REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	26.Oct.98	MAJOR REPORT

4. TITLE AND SUBTITLE
IRIDIUM TTC GROUND STATION NUMBER & LOCATION OPTIMIZATION

5. FUNDING NUMBERS

6. AUTHOR(S)
2D LT MASINO AARON J

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UNIVERSITY OF COLORADO AT COLORADO SPRINGS

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
98-015

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Iridium TTC Ground Station Number and Location Optimization

1 May 1998
Aaron J. Masino
ME Space Operations

Table of Contents

Abstract

The commercialization of space missions has led to the need for cost minimization in order to make these ventures commercially viable. In the past, commercial space systems have generally consisted of a single satellite in a geo-stationary orbit. Command and control of these systems required only a single ground station and cost was not an issue. However, with the crowding of the geo-stationary belt and the desire for truly global wireless communication, companies have now begun developing mission concepts using proliferated low Earth orbit (LEO) satellite systems. These systems cannot be commanded by a single ground station. Therefore, in order to minimize cost for the system, the minimum number of ground stations required for command and control operations must be determined. This paper develops a method for determining the minimal number of ground stations required to command and control Motorola's Iridium satellite constellation during the mission phase. The method relies on the use of the Iridium satellites' ability to communicate with other satellites in the system, that is the satellites' cross-link abilities. Using a computer model, the method will determine which satellite is currently in view of a given "candidate" ground station position. Then, based on delay time for ground station processing and satellite processing, the method will determine how many satellites the current ground station could command using satellite cross-links. It does this for every latitude and longitude position until it finds the position that can command the most satellites in the system. From there, the method determines the next ground station position in the same manner, until all the satellites have been commanded. Limitations of the given method as well as generalizations that can be made to the method in order to determine the minimal number of ground stations required for normal operations of any proliferated LEO satellite system are discussed.

Nomenclature

- D is the total time necessary for a ground station to command a given satellite
- p is the amount of time that a ground station's request for information or a satellite's response to that request waits in the transmission queue
- x is the command load processing time on the ground, that is the amount of time that it takes a ground station to generate a command load for a given satellite once it has received health and safety data on that satellite
- y is the ground station processing time, that is the amount of time that it takes for a ground station to process a satellites health and safety information before it begins generating a command load
- z is the satellite processing time, that is the amount of time it takes to process and package health and safety information before transmission, once it has acted on a request for information
- α is the number of cross-links that must be used in order for a ground station communicating with satellite A to communicate with satellite B
- β is the number of cross-links that satellite B must use in order to reach a ground station which is communicating with satellite A
- ε the time at which for one satellite pass over a ground station the geometry of the problem changes, that is the satellites maintaining left and right cross-links change (explained below)

Background

Motorola's Iridium project is a proliferated low Earth orbit (LEO) satellite system that is designed to provide global wireless communication needs such as voice, paging, data and facsimile transmissions.[1] The system works much like a cellular phone. The major difference between a cellular phone and an Iridium phone is that, unlike a cellular phone, an Iridium phone can be used anytime, anywhere in the world. Instead of relying strictly on cellular towers, an Iridium phone can, when a cellular tower is unavailable, access a satellite. Calls made from one location on the Earth are relayed via satellite cross-links to another part of the Earth. The system works on two levels. A user phone first attempts to access a local cellular tower. If a cellular tower is available, the user phone will operate using the cellular tower in the same manner as a cellular phone. If a cellular tower is unavailable, the user phone will access a satellite. The satellite will then route the call through a path from itself to a satellite that is over the location of the second user. The second user does not have to be using an Iridium phone. A satellite transmits calls to the ground where gateways relay calls from the satellite to standard telephone lines.[1]

The system consists of 66 satellites. The satellites are in 6 orbital planes with 11 operational satellites per plane. The satellites are in near circular orbits at an altitude of 780 kilometers with a period of 100 minutes 27 seconds.. The planes are inclined at 86.4 degrees for a near polar orbit.[2] This orbital configuration provides 100 percent coverage of the Earth's surface.

fig 1. Iridium orbital planes

The satellites themselves are equipped with an extensive communications package. Transmissions are made to the ground using 48 spot beams with a link margin of 16 dB for voice communications. [1] Each satellite is equipped with four additional antennas for cross-link purposes. Satellites are capable of directly communicating with four other satellites. Any given satellite can transmit information to and receive information from the satellite in front of and behind it in the same orbital plane and the satellites to its immediate right and left in the adjacent planes. However, as the satellites approach the poles, the cross-links to the satellites in the adjacent planes are lost. However, the links within the same plane are maintained.[3] The particular latitude of this cross-link outage will be discussed below.

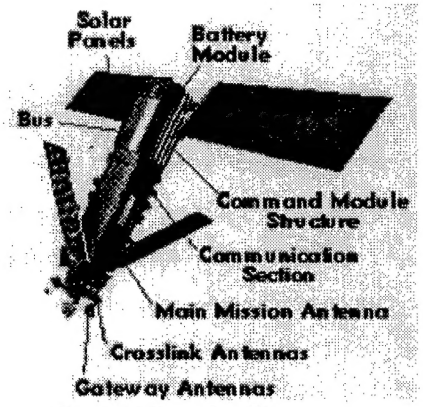


fig. 2 Iridium Satellite

Ground stations are needed to operate and maintain these satellites. Operating and maintaining a satellite involves several functions. The state of health of every satellite has to be

monitored. Health and safety data includes information on all the subsystems of the satellite, such as the communications, power and thermal systems. Depending on the complexity of the satellite, this data can grow quite large. For the Iridium system, the health and safety of each satellite is a relatively small amount of data. The only systems that are critical are power and the communications payload.[4] However, upon review of health and safety information, ground crews must handle anomaly resolution. If a satellite is operating outside of its normal parameter range, the satellite must be "safed" until the ground crew develops a command load that addresses the anomaly. These commands must then be uploaded to the satellite. Once the commands are uploaded, the satellite must be monitored further to ensure that the command load worked as planned. If the command load failed to resolve the problem, the whole process must be repeated. In a system of 66 satellites, resolving anomalies, which are sometimes frequent, could potentially become a major operations crisis. In a commercial system, the system availability must be extremely high, therefore, anomalies must be resolved quickly.

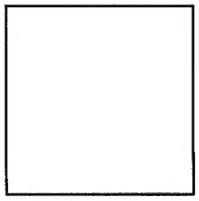
An even larger problem for Iridium operators will be maintaining the integrity of the constellation. Orbit management will be critical to the Iridium project. As the satellites tend to drift from their original positions, operators must, on a regular basis, conduct ranging on each satellite to determine orbital parameters and move the satellites into their proper orbital slots. Time keeping will also be crucial to the system. Because timing for the constellation is critical to a digital communications system, a master clock must be maintained at an operations center. This clock must then be used to constantly update each satellite clock. For a system in a LEO orbit, such as Iridium, the accuracy of the satellite clocks must be monitored frequently. Finally, there is the issue of network management. Because the satellites are moving so quickly, the pattern of cross-links used to route a transmission from one satellite to another is constantly

changing. Although the Iridium satellites each have on-board routing software to conduct this function, the integrity of the software must be monitored by the ground stations. Should there be a problem with this software on any satellite, that problem must be resolved almost immediately. If it is not, there will be lost transmissions on a regular basis which is not acceptable in a commercial system.

Introduction

The goal of this study is to determine the minimal number of ground stations required to command and monitor health and safety information for the entire Iridium constellation within the time of a single satellite pass over each of the ground stations. Although this time limit is somewhat strict, it is realistic. These satellites must be working properly in order for the system to work. System failure leads to lost customers. For a system that has already cost over 6 billion dollars, a loss of customers is not an option.[2]

The number of ground stations required to meet this standard will be based on different values of total time to command a satellite, D, where:



$$D = (x + y + z) + (\alpha + \beta + 2)p \qquad (1)$$

where x=2 seconds, command load processing time
y=5 seconds, ground station processing time
z=1 second, satellite processing time
α and β are defined in the nomenclature

Although the actual values of these parameters are currently unavailable from Motorola, the estimated values are realistic because the satellites are relatively simple and the command

loads are of a standard form.[4] If there is an anomaly, a satellite will automatically be sent a safe command. The ground crew will attempt to develop a solution to the anomaly. Once the solution command load has been developed, it will be sent to the afflicted satellite using the satellite that has currently moved into view. This allows a ground crew to command as many satellites as is possible in one satellite pass without spending the entire pass time attempting to resolve an anomaly.

Because x, y and z are constant the only varying factor in equation 1 is the amount of time that a ground station request or a satellites reply to a request waits in the satellite's transmission queue, p. When a ground station sends a request for information, that request is placed into the same queue as customer transmissions. When a satellite collects its health and safety information and packages a response to the ground station, that response is also placed into the transmission queue. This transmission queue uses a priority system. Customer transmissions have a higher priority than health and safety requests and replies.[3] Therefore, the higher the amount of customer activity the longer a request or reply will wait for transmission on-board any given satellite. The value of p will therefore be set at different values so that the number of ground stations required will be based on a given value of p. The value of p will be assumed to be a constant for all satellites for any given ground station number determination. This is a limitation, however, because a more accurate model would change p based on each satellite's current location. This is true because a satellite over North America for example, is most probably going to have a higher value of p than one currently over the Pacific ocean due to an obviously higher number of users in the region of North America than in the Pacific ocean. However, expected data loads, and resulting delay times for health and safety requests/replies, by region are unavailable from Motorola, thus the given development of p will be used.

The approach of this paper will be to first conduct an orbital analysis. This is necessary to determine a mean time in view. It is also necessary so that the arrangement, or geometry, of the satellite constellation during a ground station pass can be understood. As discussed below, an understanding of the geometry of the problem will greatly ease the resulting algorithm to determine the required number of ground stations.

Once the orbital analysis is done, the next step is to determine the number of ground stations required for a given value of p. This will be discussed in detail below, but the general method is to cycle through all latitude and longitude combinations for the first ground station. Using the mean time in view and the current constellation geometry determine which satellite is closest to that latitude longitude combination. If it is assumed that no ground stations will be placed above 65 North latitude or below 65 degrees South latitude, the worst case will be that only one satellite will be in view of the ground station but there will always be at least one. That is, after all, the point of the Iridium project. The requirement that ground stations not be above 65 N or below 65 S forces the worst case of having only one satellite in view. Above these latitudes a ground station would always have access to more than one satellite. However the only land below 65 S latitude is Antarctica and the only land above 65 N latitude are the extreme northern regions of Canada, Greenland and Russia. All of these areas are assumed to be out of use due to practical considerations of weather and expense. Using this satellite and the current delay time, determine the number of satellites that this particular latitude longitude combination could command and control. Doing this for each latitude and longitude combination, find the one that can command the most satellites. Continue doing this for successive ground stations until all satellites have been commanded. This, however, may result in an optimal ground station

being placed in a latitude longitude combination that is located over water or in a politically unstable region. Therefore the optimal ground stations may have to be altered.

The software that has been developed is capable of first giving the optimal ground station locations. Then, once the user has modified the locations to correct for water and political considerations, the software is capable of determining whether or not these locations will be able to command all of the satellites. If not, at least one more ground station will be required to command and control the entire constellation. A limitation of the software is that it assumes that the orbital analysis has been conducted previously. Thus when used, the software must be given the mean time in view and the satellite constellation geometry, as is discussed below.

Orbital Analysis

In order to determine a mean time in view and to develop an understanding of the constellation geometry, an extensive orbital analysis was conducted using Satellite Tool Kit from Analytical Graphics, Inc. The first required piece of information was a mean time in view for a satellite pass over a ground station. This was done by placing a satellite into an Iridium orbit. The satellite was then fixed with a communications sensor. The sensor was given an outer half angle of 70 degrees. Thus the sensor did not use a spot beam but a semi-omni directional type antenna. Then a ground station facility was place in three positions, one at 55 degrees south latitude, one at 0 degrees latitude and one at 55 degrees north latitude. A communications sensor was located at each ground station. The minimum elevation angle was set at three different angle, 10, 15 and 20 degrees. The chains tool within Satellite Tool Kit was then used to create a link between the satellite antenna to each of the ground station antennas at each elevation angle. Finally, the reports tool was used to create a report on the access times between the satellite antenna and the ground station antenna for each ground station at each elevation angle. The

reports are listed in appendix 1 and are for a period of 10 days. The nine different mean duration times where then averaged to arrive at a mean time in view of:

$$\frac{5155.113}{9} = 572.790\bar{3}\,\text{sec} = 9.55\,\text{min} \quad (2)$$

where the 9 mean times in view are:

Latitude	Elevation (deg)	Mean Time In View(seconds)
0	10	597.085
0	15	554.248
0	20	521.518
55 North	10	626.954
55 North	15	600.532
55 North	20	579.44
55 South	10	609.899
55 South	15	557.981
55 South	20	507.459

The next step was to determine the latitude at which the satellite's left and right cross-links stop working. The exact value is Motorola proprietary information and thus had to be determined using Satellite Tool Kit. This was done by forming an Iridium constellation and placing a sensor on two satellites in adjacent planes. The sensors were set to fixed targeting so that they were intended to communicate with each other only. Next, the chains tool was again used to connect the two satellite antennas. Finally, a report was generated to determine the contact period of the satellite starting at 0 degrees latitude and ending at 90 degrees latitude. The length of the contact period was 1211.274 seconds. The report is given in appendix 2. Assuming a spherical Earth and knowing the period of an Iridium satellite, it is possible to determine a ratio of degrees latitude covered per second as:

$$\frac{360}{6027.14} = .059729822\,\frac{\text{deg}}{\text{sec}}$$

Now, knowing the length of the contact period and the fact that the contact period began at 0 latitude and that this period was continuous, it is possible to determine the latitude at which the cross-link was lost as:

$$lat = 90 - (1506.785 - 1211.274).059729822$$
$$lat = 72.349 \deg \quad (3)$$

where 1506.785 is ¼ of the satellites period in seconds, that is the time required for the satellite to travel from 0 degrees latitude to 90 degrees latitude.

The final result needed from the orbital analysis was a development of the constellation geometry. The constellation is remarkably well behaved. At any one time there are always 3 satellites in each plane that are without left and right cross-links. The geometry is constant in that there are always two satellites in each plane that are either above 72 N latitude or below 72 degrees S latitude and one satellite at the opposite pole. The remaining 8 satellites in each plane are evenly spaced between 72 S and 72 N latitude. So that for any given longitude and latitude combinations below 65 N or above 65 S a ground station will have access to the same satellite in one pass for the stated mean time in view and for that pass there will be exactly 8 satellites in each plane that can maintain cross-links. However the geometry does change once during this pass. From the start of the satellite pass over a ground station the same 8 satellites in each orbital plane maintain cross-links until a time ε. At ε, the most northerly satellites moving in ascending motion that had maintained left and right cross-links, lose them. At the same time the most southerly satellites moving in descending motion lose their left and right cross-links. However at this same time, ε, new satellites descend below 72 N latitude and ascend above 72 S latitude so that there are still 8 satellites maintaining left and right cross-links. If a satellite pass

is said to begin at time zero, then ε occurs at

$$\varepsilon = 3\,\text{min}\,27\,\text{sec}$$

into the satellite pass.

The importance of this geometry stability will become evident below in the discussion of the algorithm used to solve for the minimal number of ground stations required.

Method

Once the geometry has been determined, the method for determining the minimum number of ground stations using computer software follows. The software uses the fact that for any satellite pass over a ground station the geometry will be the same. As stated, the satellite will access one of the four satellites in a plane that maintains left and right cross-links. There will always be 8 satellites per plane that are capable of left and right cross-links. Further, there will be only one change in geometry in the pass, at time ε, as stated. The fact that the geometry does not change allows this problem to be treated statically. Using Satellite Tool Kit, the initial latitude and longitude of the sub-satellite positions can be determined. The next step is to number the satellites in each plane 1 through 11. Satellite Tool Kit will do this automatically. The final step is to determine the numbered satellites that lose left and right cross-links for the pre-epsilon time and the post-epsilon time, that is which satellites are above 72 N and below 72 S. The satellites will be the same for each plane. For example, if satellite 1 in plane 1 has lost left and right cross-links, then satellite 1 in every other plane will have lost left and right cross-links. Appendix 4 has a picture of the ground track.

From Appendix 4, it is seen that for the Iridium setup, the satellites that have lost left and right cross-links before ε are satellites 3, 8, and 9 in each plane. The next step is to determine

which satellites have lost left and right cross-links after ε. For this case, satellites 2, 3 and 8 in each plane have lost their left and right cross-links.

At this point the computer algorithm is ready to begin. The program will first query the user for information on the cross-links. It will ask for the number of the satellites that have lost cross-links before and after ε and for the transitional case. So for this case the satellites that have lost left and right cross-links before ε are entered as 3, 8 and 9. The satellites that have lost left and right cross-links after ε will be entered as 2, 3 and 8. Finally, the transitional case, the satellite numbers will be entered as 2, 3, 8 and 9. The transitional case will be explained below.

The cross-link information is used by the software to create a computer graph. A graph is defined as a set of vertices and arcs. An arc between vertex a and vertex b means that it is possible to travel directly from a to b and from b to a.[5] For this software, the vertices are the satellites and an arc from satellite A to B represents a direct cross-link between satellites A and B. The graph will be represented by an adjacency matrix. An adjacency matrix is an n x n matrix. In this case, the rows and columns of the matrix are each of the satellite labels. The satellites are labeled 0 to 65. The reason for beginning at 0 and not 1 is that C++ sets array (i.e. matrix) origins at (0,0). Entries in the matrix are Boolean values, represented by 0 and 1. A one value in the position i,j in the matrix indicates that there is a cross-link between satellites i and j. By convention, a vertex is always considered to be connected to itself, thus all positions i,i will be given a value of 1.[5] Using this method and the cross-link information entered by the user, the software develops three adjacency matrices. The first represents the graph cross-link information before ε. The second represents the graph cross-link information after ε. The third is used for the special case of when satellite A is attempting to send a message to satellite B using cross-links and the time ε passes before the message arrives at B but has already left A. That is,

when satellite A originally transmits the message to some intermediate satellite, it does so before ε, so that the pre-epsilon geometry is correct. However while the message for satellite B is at an intermediate satellite, ε passes. At this point, the post-epsilon geometry takes over. However determining the exact satellite location of this geometry change makes the problem too dynamical. It is more consistent to assume an intermediate geometry in which four satellites in each plane have lost cross-links, those being the satellites 2, 3, 8 and 9 for this case. Because it can be determined a priori if a transmission will be subject to this condition, the third adjacency matrix can be used for this case.

Once the program has formed these adjacency matrices it must next read the satellite latitude and longitude information. These are the values determined by Satellite Tool Kit and are read in from a file. The program will ask the user for the filenames that contain the latitude and longitude information. All that is required of the user is to enter the names.

The software is now ready to compute the number of ground stations required. It does so by first choosing a candidate latitude and longitude position for a ground station. The software considers positions at 5 degree increments, although this could easily be modified to different increments by change the step size of the loop counters.

It next finds the satellite that is closest to the current ground station location. It does this by comparing the ground station longitude and latitude to each satellite longitude and latitude. The satellite with the smallest absolute difference is chosen as the root satellite, meaning that this satellite will act as the one in view of these coordinates for the satellite pass.

The software next finds the shortest path between the root satellite and every other satellite. It does this using Dijkstra's Algorithm, as described by Budd, which is a single source shortest path algorithm.[5] Dijkstra's Algorithm is used to determine the number of cross-links,

α, required to reach a given satellite from the root satellite. With this information, the request delay time is determined as:

$$request = (\alpha + 1)p + z \tag{4}$$

The adjacency matrix used is determined by αp. If this value is less than ε, then the pre-epsilon matrix is used, otherwise the special case matrix is used. Dijkstra's Algorithm is then used to determine the shortest path from a given satellite to the root satellite. It will return the number of cross-links, β, required to reach the root satellite from the current satellite. If

$$request + x + y \leq \frac{\varepsilon}{2}$$

then the pre-epsilon matrix is used. If

$$\frac{\varepsilon}{2} < request + x + y < \varepsilon$$

the special case matrix is used. Finally if

$$request \geq \varepsilon$$

the post-epsilon matrix is used. The respond delay time is computed by

$$respond = (\beta + 1)p \tag{5}$$

The total delay time is composed of the delay time for a ground station request to reach the satellite and the delay time for the satellite response to reach the ground station. So total delay time is given by:

$$D = request + response + x + y \tag{6}$$

It is assumed that if the ground station transmits the command load before the satellite pass is over than that satellite has been commanded. That means that the root satellite may move out of

view of the ground station before the command load has reached the destination satellite as long as the ground station transmits the command load to the root satellite while it is still in view. If

$$D \leq mean\_time\_in\_view$$

then the satellite is tagged as having been commanded by the current ground station. This is done for each satellite at each ground station location. The ground station that commands the most satellites is then chosen as the first ground stations. The whole process is now repeated to find the second satellite until all satellites have been commanded by the set of ground stations. Note that after each ground station is determined the satellites that it commands are permanently tagged as having been commanded. This is done so that when the successive ground station is chosen it only considers satellites that haven't been commanded as contributing to the new ground stations commanding ability. However, when attempting to access the remaining satellites, the new ground station still has access to all cross-links because it still uses the original three adjacency matrices.

Once the final set of ground stations has been determined the program returns the number of ground stations required as well as their latitude and longitude positions. The code listing is given in Appendix 3.

Results

The basic result is that as p increases the required number of ground stations also increases (Appendix 5). The most important values are those for p between 1 and 56 seconds. If p is 30 seconds or less the entire constellation can be commanded with one ground station. This ground station can be located at any latitude and longitude. By appendix 5 the optimal value of the ground station location is 65 N latitude and 180 longitude. This determination, however, is due to the way that the computer algorithm selects a ground station. The first latitude and

longitude combination that it considers is 65 N latitude and 180 longitude. Because this ground station succeeds in commanding all of the ground stations the algorithm does not consider others. However using the subroutine, test_gs, the user can test certain ground station locations to determine if those locations will command all of the satellites for the given value of p. Using this subroutine it has been determined that for p less than or equal to 30 seconds the single ground station can be located at any latitude and longitude combination. At p equal to 35 seconds, the required number of ground stations increases to 2. Two ground stations are required for p between 35 and 56 seconds. The optimal locations for each value of p in this range is given in Appendix 5. Once p increases beyond 56 seconds 4 or more ground stations are required to command the constellation as listed in Appendix 5. The subroutine test_gs can again be used on these values to determine whether or not adjustments to these locations will still command the entire constellation. For example, if p is set to 45 seconds then two ground stations in the United States at 40 latitude and -80 and -120 longitude can command the entire constellation. However for p greater than or equal to 50 seconds no two ground stations on land can command the entire constellation. Therefore the satellite capacity would have to be designed to be less than 46 seconds to have to land based ground stations. Note also from Appendix 5 that at no value of p are 3 ground stations required. At 57 seconds, the required number of ground stations is 4. The Iridium system currently calls for 3 ground stations. However, as it has been shown, only two are necessary. This reduction will greatly reduce costs because one less ground station is required and therefore fewer operators are required.

Limitations

There are certain limitations to this approach. The first and most important is the restriction that the number of ground stations determined must be capable of commanding all

satellites for a single mean pass time. This restriction forces the determined number of ground stations to have the capability to constantly command and control all of the satellites. This is true because as the current root satellite at each ground station move out of view a new root satellite will be moving into view. Other approaches might restrict the determined number of ground stations as having the ability to only command all of the satellites once every orbital period, once every 8 hours, once every day, etc. Such an approach would require long term knowledge of the constellation geometry. Each satellite that comes into view of a ground station during the given period would have to be determined for each ground station location. For each of those satellites the in-view period would have to be determined. Finally using each in view period, the particular satellites that could be commanded from that ground station location would have to be determined. This would have to be done for each ground station candidate location. The process is similar to that given in this paper. However, the need for long term knowledge of the orbit in order to determine each satellite that will come into view of each ground station location makes the problem extremely dynamical. Furthermore, as stated above, because this is a commercial venture, that requires that all satellites be working properly, if they are monitored only periodically problems may arise. For example, if every satellite is commanded only once every 4 hours, a given satellite suffering from an anomaly could be down more than 4 hours. If only a few satellites are down for a few hours at a time, system performance will degrade dramatically.

The other limitation is the case for which a transmission is made from satellite A to satellite B using the special case adjacency matrix. This case has assumed that for the entire path from A to B that there are four satellites in each plane that have lost left and right cross-links. The true case, however, is that for the path there will only be 3 satellites in each plane that have lost left and right cross-links. At some point along that path the satellites that lose these cross-

links changes. This approach does not attempt to determine where in the path that this change occurs, but rather assumes that for the entire path the cross-links lost in the pre-epsilon and post-epsilon geometry have been lost for this particular case. Another method could determine the location along the path that the geometry changes and compute the remaining path from the intermediate satellite to the destination satellite using the post-epsilon geometry.

Applications

The given method has two applications. It can be used to determine the minimal number of ground stations required for any proliferated LEO constellation. Certain modifications would, however, have to be made to the developed software. An orbital analysis would be required of the given constellation to determine the parameters required as discussed above. The software would have to be modified to account for an increase or decrease in the number of orbital planes, number of satellites per plane and the total number of satellites in the constellation. Secondly, the software would have to be modified to fit the cross-link capability of the given system. For example if the new system had more or less than 4 cross-links per satellite this must be corrected in the software. The software would also have to modified to account for which satellites in each plane may have lost certain cross-link capabilities. The last modification would have to account for more than one geometry change in a satellite pass over the ground station. For each time, or ε, at which the geometry changes a new adjacency matrix must be determined. This new matrix would then be used as described above to determine the path from satellite A to satellite B.

Another application of this method is a by product of the shortest path algorithm. With the use of knowledge of the constellation orbit and timers, an adjacency matrix for each satellite at each time period could be determined. Once done this could be used at the satellites routing software. For a satellite attempting to relay a call from one location on Earth to another, it could

use the current adjacency matrix to determine the shortest path from its location to the destination location. This combined with a knowledge of which satellites are currently in safe-mode and are therefore unavailable to the path could provide efficient routing software. Currently each Iridium satellite has on-board routing software, but the nature of this software is Motorola proprietary information and is therefore unavailable.[3]

Conclusion

Commercialization of space operations requires efficient operations at low costs. This requirement forces commercial developers to determine the minimal number of ground stations required to command and control their space assets. This paper has discussed a method for determining the minimal number of ground station required to command and control Motorola's Iridium constellation. It also discusses how to apply this method to other cross-linked LEO constellations as well as certain limitations of this method.

References

1. http://alishaw.sscf.ucsb.edu/~sung/comm115/Iridium.html, "Iridium (Motorola)".

2. http://www.mot.com/GSS/SSTG/SATCOM/FAQ_A.html, "System Information".

3. Howe, Jim. Senior Spacecraft Analyst, Allied Signal Technical Services. E-mail correspondence, 3 Mar 1998.

4. Mullikin, Thomas L. Sole Proprietor, Aerospace Consulting. E-mail correspondence, 25 Feb 1998.

5. Budd, Timothy A. Classic Data Structures in C++. New York: Addison-Wesley Publishing Co., 1994.

Appendix 1

Latitude 0 deg Elevation 15
11 Apr 1998 14:04:33
Chain-sat_ground_lat0: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 04:05:06.94	1 Jan 1997 04:14:07.63	540.695
2	1 Jan 1997 05:44:23.70	1 Jan 1997 05:55:38.02	674.322
3	1 Jan 1997 16:40:26.69	1 Jan 1997 16:51:45.70	679.011
4	1 Jan 1997 18:22:04.69	1 Jan 1997 18:30:53.53	528.831
5	2 Jan 1997 03:34:01.61	2 Jan 1997 03:38:35.05	273.434
6	2 Jan 1997 05:10:22.39	2 Jan 1997 05:22:16.95	714.562
7	2 Jan 1997 06:56:24.43	2 Jan 1997 06:58:45.64	141.215
8	2 Jan 1997 16:07:37.61	2 Jan 1997 16:17:07.87	570.264
9	2 Jan 1997 17:47:23.42	2 Jan 1997 17:58:23.78	660.357
10	3 Jan 1997 04:36:53.86	3 Jan 1997 04:48:25.58	691.726
11	3 Jan 1997 06:19:09.94	3 Jan 1997 06:27:19.72	489.779
12	3 Jan 1997 15:35:37.23	3 Jan 1997 15:41:19.79	342.566
13	3 Jan 1997 17:13:18.27	3 Jan 1997 17:25:10.48	712.205
14	4 Jan 1997 04:04:02.38	4 Jan 1997 04:14:01.61	599.230
15	4 Jan 1997 05:44:08.22	4 Jan 1997 05:54:49.32	641.097
16	4 Jan 1997 16:39:43.63	4 Jan 1997 16:51:23.86	700.221
17	4 Jan 1997 18:22:07.44	4 Jan 1997 18:29:35.86	448.415
18	5 Jan 1997 03:32:11.65	5 Jan 1997 03:38:53.39	401.741
19	5 Jan 1997 05:09:53.46	5 Jan 1997 05:21:40.40	706.937
20	5 Jan 1997 16:06:40.76	5 Jan 1997 16:17:02.26	621.505
21	5 Jan 1997 17:47:08.72	5 Jan 1997 17:57:30.44	621.724
22	6 Jan 1997 04:36:12.61	6 Jan 1997 04:48:00.42	707.814
23	6 Jan 1997 06:19:29.10	6 Jan 1997 06:26:06.34	397.244
24	6 Jan 1997 15:34:18.95	6 Jan 1997 15:41:48.52	449.574
25	6 Jan 1997 17:12:51.03	6 Jan 1997 17:24:31.50	700.465
26	7 Jan 1997 04:03:05.85	7 Jan 1997 04:13:49.49	643.639
27	7 Jan 1997 05:43:59.12	7 Jan 1997 05:53:55.24	596.119
28	7 Jan 1997 16:39:05.05	7 Jan 1997 16:50:57.13	712.085
29	7 Jan 1997 18:22:19.97	7 Jan 1997 18:27:59.94	339.967
30	8 Jan 1997 03:30:46.65	8 Jan 1997 03:39:00.39	493.743
31	8 Jan 1997 05:09:29.28	8 Jan 1997 05:20:59.48	690.199
32	8 Jan 1997 16:05:49.60	8 Jan 1997 16:16:49.64	660.040
33	8 Jan 1997 17:46:59.70	8 Jan 1997 17:56:29.91	570.208
34	9 Jan 1997 03:00:38.50	9 Jan 1997 03:03:07.40	148.901
35	9 Jan 1997 04:35:36.12	9 Jan 1997 04:47:30.90	714.777
36	9 Jan 1997 06:20:13.50	9 Jan 1997 06:24:41.53	268.023
37	9 Jan 1997 15:33:10.36	9 Jan 1997 15:41:59.56	529.197
38	9 Jan 1997 17:12:28.26	9 Jan 1997 17:23:47.58	679.326

Global Statistics

Min Duration	7	2 Jan 1997 06:56:24.43	2 Jan 1997 06:58:45.64	141.215
Max Duration	35	9 Jan 1997 04:35:36.12	9 Jan 1997 04:47:30.90	714.777
Mean Duration			554.241	
Total Duration			21061.156	

Latitude 0 degrees Elevation 10 degrees
11 Apr 1998 14:08:17
Chain-sat_ground_lat0: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 04:05:06.94	1 Jan 1997 04:14:51.54	584.607
2	1 Jan 1997 05:44:23.70	1 Jan 1997 05:56:27.98	724.276
3	1 Jan 1997 16:39:36.58	1 Jan 1997 16:51:45.70	729.125
4	1 Jan 1997 18:21:21.40	1 Jan 1997 18:30:53.53	572.121
5	2 Jan 1997 03:34:01.61	2 Jan 1997 03:38:59.48	297.873
6	2 Jan 1997 05:10:22.39	2 Jan 1997 05:23:08.15	765.763
7	2 Jan 1997 06:56:24.43	2 Jan 1997 06:58:56.16	151.729
8	2 Jan 1997 16:06:51.93	2 Jan 1997 16:17:07.87	615.945
9	2 Jan 1997 17:46:34.21	2 Jan 1997 17:58:23.78	709.570
10	3 Jan 1997 04:36:53.86	3 Jan 1997 04:49:15.94	742.082
11	3 Jan 1997 06:19:09.94	3 Jan 1997 06:28:01.10	531.157
12	3 Jan 1997 15:35:06.39	3 Jan 1997 15:41:19.79	373.405
13	3 Jan 1997 17:12:27.19	3 Jan 1997 17:25:10.48	763.284
14	4 Jan 1997 04:04:02.38	4 Jan 1997 04:14:48.33	645.943
15	4 Jan 1997 05:44:08.22	4 Jan 1997 05:55:38.06	689.836
16	4 Jan 1997 16:38:52.84	4 Jan 1997 16:51:23.86	751.020
17	4 Jan 1997 18:21:28.90	4 Jan 1997 18:29:35.86	486.956
18	5 Jan 1997 03:32:11.65	5 Jan 1997 03:39:28.71	437.057
19	5 Jan 1997 05:09:53.46	5 Jan 1997 05:22:31.40	757.936
20	5 Jan 1997 16:05:52.80	5 Jan 1997 16:17:02.26	669.463
21	5 Jan 1997 17:46:21.04	5 Jan 1997 17:57:30.44	669.403
22	6 Jan 1997 04:36:12.61	6 Jan 1997 04:48:51.34	758.735
23	6 Jan 1997 06:19:29.10	6 Jan 1997 06:26:41.54	432.437
24	6 Jan 1997 15:33:40.08	6 Jan 1997 15:41:48.52	488.438
25	6 Jan 1997 17:12:00.37	6 Jan 1997 17:24:31.50	751.128
26	7 Jan 1997 04:03:05.85	7 Jan 1997 04:14:38.06	692.207
27	7 Jan 1997 05:43:59.12	7 Jan 1997 05:54:42.11	642.991
28	7 Jan 1997 16:38:13.90	7 Jan 1997 16:50:57.13	763.227
29	7 Jan 1997 18:21:49.51	7 Jan 1997 18:27:59.94	370.426
30	8 Jan 1997 03:30:46.65	8 Jan 1997 03:39:41.73	535.079
31	8 Jan 1997 05:09:29.28	8 Jan 1997 05:21:49.96	740.683
32	8 Jan 1997 16:05:00.15	8 Jan 1997 16:16:49.64	709.489
33	8 Jan 1997 17:46:14.33	8 Jan 1997 17:56:29.91	615.585
34	9 Jan 1997 03:00:38.50	9 Jan 1997 03:03:18.73	160.237
35	9 Jan 1997 04:35:36.12	9 Jan 1997 04:48:22.08	765.958
36	9 Jan 1997 06:20:13.50	9 Jan 1997 06:25:05.54	292.036
37	9 Jan 1997 15:32:26.76	9 Jan 1997 15:41:59.56	572.800
38	9 Jan 1997 17:11:38.34	9 Jan 1997 17:23:47.58	729.239

Global Statistics

Min Duration	7	2 Jan 1997 06:56:24.43	2 Jan 1997 06:58:56.16	151.729
Max Duration	35	9 Jan 1997 04:35:36.12	9 Jan 1997 04:48:22.08	765.958
Mean Duration			597.085	
Total Duration			22689.243	

Latitude 0 degrees Elevation 20 degrees
11 Apr 1998 14:10:25
Chain-sat_ground_lat0: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 04:05:06.94	1 Jan 1997 04:13:34.73	507.792
2	1 Jan 1997 05:44:23.70	1 Jan 1997 05:54:58.32	634.618
3	1 Jan 1997 16:41:06.60	1 Jan 1997 16:51:45.70	639.105
4	1 Jan 1997 18:22:36.94	1 Jan 1997 18:30:53.53	496.591
5	2 Jan 1997 03:34:01.61	2 Jan 1997 03:38:19.73	258.119
6	2 Jan 1997 05:10:22.39	2 Jan 1997 05:21:35.61	673.221
7	2 Jan 1997 06:56:24.43	2 Jan 1997 06:58:39.81	135.376
8	2 Jan 1997 16:08:12.30	2 Jan 1997 16:17:07.87	535.572
9	2 Jan 1997 17:48:02.35	2 Jan 1997 17:58:23.78	621.426
10	3 Jan 1997 04:36:53.86	3 Jan 1997 04:47:45.26	651.400
11	3 Jan 1997 06:19:09.94	3 Jan 1997 06:26:49.58	459.637
12	3 Jan 1997 15:35:57.65	3 Jan 1997 15:41:19.79	322.147
13	3 Jan 1997 17:13:59.48	3 Jan 1997 17:25:10.48	670.991
14	4 Jan 1997 04:04:02.38	4 Jan 1997 04:13:25.61	563.230
15	4 Jan 1997 05:44:08.22	4 Jan 1997 05:54:11.10	602.884
16	4 Jan 1997 16:40:24.42	4 Jan 1997 16:51:23.86	659.433
17	4 Jan 1997 18:22:34.87	4 Jan 1997 18:29:35.86	420.983
18	5 Jan 1997 03:32:11.65	5 Jan 1997 03:38:29.00	377.344
19	5 Jan 1997 05:09:53.46	5 Jan 1997 05:20:59.35	665.882
20	5 Jan 1997 16:07:18.04	5 Jan 1997 16:17:02.26	584.221
21	5 Jan 1997 17:47:45.83	5 Jan 1997 17:57:30.44	584.611
22	6 Jan 1997 04:36:12.61	6 Jan 1997 04:47:19.40	666.792
23	6 Jan 1997 06:19:29.10	6 Jan 1997 06:25:42.12	373.022
24	6 Jan 1997 15:34:46.60	6 Jan 1997 15:41:48.52	421.917
25	6 Jan 1997 17:13:31.74	6 Jan 1997 17:24:31.50	659.761
26	7 Jan 1997 04:03:05.85	7 Jan 1997 04:13:11.33	605.480
27	7 Jan 1997 05:43:59.12	7 Jan 1997 05:53:19.21	560.089
28	7 Jan 1997 16:39:46.30	7 Jan 1997 16:50:57.13	670.833
29	7 Jan 1997 18:22:40.11	7 Jan 1997 18:27:59.94	319.830
30	8 Jan 1997 03:30:46.65	8 Jan 1997 03:38:30.18	463.533
31	8 Jan 1997 05:09:29.28	8 Jan 1997 05:20:19.10	649.821
32	8 Jan 1997 16:06:28.68	8 Jan 1997 16:16:49.64	620.965
33	8 Jan 1997 17:47:34.20	8 Jan 1997 17:56:29.91	535.707
34	9 Jan 1997 03:00:38.50	9 Jan 1997 03:03:01.06	142.562
35	9 Jan 1997 04:35:36.12	9 Jan 1997 04:46:49.56	673.445
36	9 Jan 1997 06:20:13.50	9 Jan 1997 06:24:26.55	253.046
37	9 Jan 1997 15:33:42.80	9 Jan 1997 15:41:59.56	496.755
38	9 Jan 1997 17:13:08.04	9 Jan 1997 17:23:47.58	639.543

Global Statistics

Min Duration	7	2 Jan 1997 06:56:24.43	2 Jan 1997 06:58:39.81	135.376
Max Duration	35	9 Jan 1997 04:35:36.12	9 Jan 1997 04:46:49.56	673.445
Mean Duration			521.518	
Total Duration			19817.686	

Latitude -55 degrees Elevation 10 degrees
11 Apr 1998 14:07:39
Chain-Sat_ground_lat_neg55: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 07:44:00.95	1 Jan 1997 07:49:55.67	354.715
2	1 Jan 1997 09:20:48.79	1 Jan 1997 09:32:59.79	731.003
3	1 Jan 1997 11:01:21.57	1 Jan 1997 11:13:51.26	749.690
4	1 Jan 1997 12:45:27.55	1 Jan 1997 12:52:52.10	444.545
5	1 Jan 1997 19:44:56.05	1 Jan 1997 19:55:23.50	627.444
6	1 Jan 1997 21:24:59.90	1 Jan 1997 21:37:55.35	775.450
7	1 Jan 1997 23:06:27.08	1 Jan 1997 23:16:60.00	632.917
8	2 Jan 1997 08:47:47.01	2 Jan 1997 08:58:56.29	669.283
9	2 Jan 1997 10:27:18.72	2 Jan 1997 10:40:11.61	772.896
10	2 Jan 1997 12:10:18.05	2 Jan 1997 12:20:01.38	583.333
11	2 Jan 1997 19:11:51.45	2 Jan 1997 19:20:23.58	512.124
12	2 Jan 1997 20:51:17.48	2 Jan 1997 21:04:00.19	762.715
13	2 Jan 1997 22:32:18.84	2 Jan 1997 22:44:06.80	707.951
14	3 Jan 1997 00:16:21.65	3 Jan 1997 00:20:17.27	235.615
15	3 Jan 1997 08:15:05.79	3 Jan 1997 08:24:40.06	574.267
16	3 Jan 1997 09:53:33.61	3 Jan 1997 10:06:25.65	772.047
17	3 Jan 1997 11:35:30.10	3 Jan 1997 11:46:45.10	675.003
18	3 Jan 1997 18:39:37.36	3 Jan 1997 18:45:00.60	323.243
19	3 Jan 1997 20:17:41.81	3 Jan 1997 20:29:47.07	725.260
20	3 Jan 1997 21:58:20.49	3 Jan 1997 22:10:54.21	753.723
21	3 Jan 1997 23:40:42.69	3 Jan 1997 23:48:25.85	463.153
22	4 Jan 1997 07:42:48.76	4 Jan 1997 07:49:59.34	430.587
23	4 Jan 1997 09:20:06.25	4 Jan 1997 09:32:33.30	747.052
24	4 Jan 1997 11:01:01.89	4 Jan 1997 11:13:16.27	734.384
25	4 Jan 1997 12:45:34.54	4 Jan 1997 12:51:44.62	370.080
26	4 Jan 1997 19:44:14.90	4 Jan 1997 19:55:15.21	660.306
27	4 Jan 1997 21:24:29.41	4 Jan 1997 21:37:23.45	774.044
28	4 Jan 1997 23:06:08.39	4 Jan 1997 23:16:03.78	595.393
29	5 Jan 1997 07:11:18.92	5 Jan 1997 07:13:34.17	135.247
30	5 Jan 1997 08:46:57.23	5 Jan 1997 08:58:33.34	696.103
31	5 Jan 1997 10:26:52.23	5 Jan 1997 10:39:39.22	766.994
32	5 Jan 1997 12:10:16.21	5 Jan 1997 12:19:14.08	537.871
33	5 Jan 1997 19:11:01.98	5 Jan 1997 19:20:23.33	561.342
34	5 Jan 1997 20:50:44.63	5 Jan 1997 21:03:34.98	770.358
35	5 Jan 1997 22:31:55.11	5 Jan 1997 22:43:18.44	683.334
36	6 Jan 1997 08:14:07.94	6 Jan 1997 08:24:22.69	614.748
37	6 Jan 1997 09:53:00.44	6 Jan 1997 10:05:55.60	775.168
38	6 Jan 1997 11:35:20.36	6 Jan 1997 11:46:04.97	644.604

39	6 Jan 1997 18:38:20.01	6 Jan 1997 18:45:09.44	409.426
40	6 Jan 1997 20:17:06.26	6 Jan 1997 20:29:28.72	742.454
41	6 Jan 1997 21:57:53.45	6 Jan 1997 22:10:13.00	739.547
42	6 Jan 1997 23:40:41.00	6 Jan 1997 23:47:16.04	395.036
43	7 Jan 1997 07:41:41.14	7 Jan 1997 07:49:53.66	492.525
44	7 Jan 1997 09:19:26.33	7 Jan 1997 09:32:05.74	759.415
45	7 Jan 1997 11:00:44.86	7 Jan 1997 11:12:40.14	715.279
46	7 Jan 1997 12:46:15.76	7 Jan 1997 12:50:09.43	233.666
47	7 Jan 1997 19:43:35.67	7 Jan 1997 19:55:04.06	688.395
48	7 Jan 1997 21:23:59.80	7 Jan 1997 21:36:49.02	769.221
49	7 Jan 1997 23:05:52.52	7 Jan 1997 23:15:04.27	551.749
50	8 Jan 1997 07:09:42.96	8 Jan 1997 07:14:36.98	294.016
51	8 Jan 1997 08:46:10.26	8 Jan 1997 08:58:08.89	718.627
52	8 Jan 1997 10:26:28.29	8 Jan 1997 10:39:05.90	757.615
53	8 Jan 1997 12:10:17.56	8 Jan 1997 12:18:22.01	484.445
54	8 Jan 1997 19:10:16.47	8 Jan 1997 19:20:19.94	603.472
55	8 Jan 1997 20:50:12.68	8 Jan 1997 21:03:07.23	774.554
56	8 Jan 1997 22:31:32.99	8 Jan 1997 22:42:27.23	654.241
57	9 Jan 1997 08:13:13.28	9 Jan 1997 08:24:02.83	649.547
58	9 Jan 1997 09:52:29.79	9 Jan 1997 10:05:24.67	774.884
59	9 Jan 1997 11:35:13.53	9 Jan 1997 11:45:22.69	609.164
60	9 Jan 1997 18:37:18.92	9 Jan 1997 18:45:14.67	475.745
61	9 Jan 1997 20:16:31.81	9 Jan 1997 20:29:07.74	755.933
62	9 Jan 1997 21:57:27.54	9 Jan 1997 22:09:29.14	721.604
63	9 Jan 1997 23:40:52.20	9 Jan 1997 23:46:01.26	309.062

Global Statistics

Min Duration	29	5 Jan 1997 07:11:18.92	5 Jan 1997 07:13:34.17	135.247
Max Duration	6	1 Jan 1997 21:24:59.90	1 Jan 1997 21:37:55.35	775.450
Mean Duration			609.899	
Total Duration			38423.613	

Latitude -55 degrees Elevation 15 degrees
11 Apr 1998 14:03:59
Chain-Sat_ground_lat_neg55: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 07:44:00.95	1 Jan 1997 07:48:37.79	276.840
2	1 Jan 1997 09:20:48.79	1 Jan 1997 09:32:07.37	678.587
3	1 Jan 1997 11:01:21.57	1 Jan 1997 11:12:55.83	694.266
4	1 Jan 1997 12:45:27.55	1 Jan 1997 12:51:14.43	346.876
5	1 Jan 1997 19:46:04.16	1 Jan 1997 19:55:23.50	559.338
6	1 Jan 1997 21:25:51.98	1 Jan 1997 21:37:55.35	723.372
7	1 Jan 1997 23:07:23.77	1 Jan 1997 23:16:60.00	576.229
8	2 Jan 1997 08:47:47.01	2 Jan 1997 08:58:01.37	614.359
9	2 Jan 1997 10:27:18.72	2 Jan 1997 10:39:18.88	720.163
10	2 Jan 1997 12:10:18.05	2 Jan 1997 12:18:47.85	509.798
11	2 Jan 1997 19:13:15.58	2 Jan 1997 19:20:23.58	427.993
12	2 Jan 1997 20:52:11.52	2 Jan 1997 21:04:00.19	708.671
13	2 Jan 1997 22:33:12.11	2 Jan 1997 22:44:06.80	654.686
14	3 Jan 1997 00:18:00.84	3 Jan 1997 00:20:17.27	136.427
15	3 Jan 1997 08:15:05.79	3 Jan 1997 08:23:40.14	514.356
16	3 Jan 1997 09:53:33.61	3 Jan 1997 10:05:34.03	720.423
17	3 Jan 1997 11:35:30.10	3 Jan 1997 11:45:42.21	612.108
18	3 Jan 1997 18:42:07.74	3 Jan 1997 18:45:00.60	172.862
19	3 Jan 1997 20:18:39.66	3 Jan 1997 20:29:47.07	667.412
20	3 Jan 1997 21:59:12.24	3 Jan 1997 22:10:54.21	701.967
21	3 Jan 1997 23:41:50.19	3 Jan 1997 23:48:25.85	395.658
22	4 Jan 1997 07:42:48.76	4 Jan 1997 07:48:49.14	360.382
23	4 Jan 1997 09:20:06.25	4 Jan 1997 09:31:41.38	695.130
24	4 Jan 1997 11:01:01.89	4 Jan 1997 11:12:19.32	677.433
25	4 Jan 1997 12:45:34.54	4 Jan 1997 12:49:43.32	248.774
26	4 Jan 1997 19:45:19.35	4 Jan 1997 19:55:15.21	595.858
27	4 Jan 1997 21:25:21.13	4 Jan 1997 21:37:23.45	722.319
28	4 Jan 1997 23:07:07.08	4 Jan 1997 23:16:03.78	536.698
29	5 Jan 1997 08:46:57.23	5 Jan 1997 08:57:39.59	642.356
30	5 Jan 1997 10:26:52.23	5 Jan 1997 10:38:45.67	713.449
31	5 Jan 1997 12:10:16.21	5 Jan 1997 12:17:54.10	457.894
32	5 Jan 1997 19:12:18.51	5 Jan 1997 19:20:23.33	484.816
33	5 Jan 1997 20:51:37.74	5 Jan 1997 21:03:34.98	717.245
34	5 Jan 1997 22:32:49.40	5 Jan 1997 22:43:18.44	629.040
35	6 Jan 1997 08:14:07.94	6 Jan 1997 08:23:25.06	557.112
36	6 Jan 1997 09:53:00.44	6 Jan 1997 10:05:03.73	723.293
37	6 Jan 1997 11:35:20.36	6 Jan 1997 11:44:58.81	578.444
38	6 Jan 1997 18:40:07.10	6 Jan 1997 18:45:09.44	302.336

39	6 Jan 1997 20:18:02.41	6 Jan 1997 20:29:28.72	686.303
40	6 Jan 1997 21:58:45.59	6 Jan 1997 22:10:13.00	687.407
41	6 Jan 1997 23:41:54.52	6 Jan 1997 23:47:16.04	321.518
42	7 Jan 1997 07:41:41.14	7 Jan 1997 07:48:48.40	427.258
43	7 Jan 1997 09:19:26.33	7 Jan 1997 09:31:14.10	707.772
44	7 Jan 1997 11:00:44.86	7 Jan 1997 11:11:41.31	656.449
45	7 Jan 1997 19:44:37.19	7 Jan 1997 19:55:04.06	626.877
46	7 Jan 1997 21:24:51.38	7 Jan 1997 21:36:49.02	717.648
47	7 Jan 1997 23:06:53.80	7 Jan 1997 23:15:04.27	490.472
48	8 Jan 1997 07:09:42.96	8 Jan 1997 07:13:10.62	207.664
49	8 Jan 1997 08:46:10.26	8 Jan 1997 08:57:16.04	665.771
50	8 Jan 1997 10:26:28.29	8 Jan 1997 10:38:11.30	703.015
51	8 Jan 1997 12:10:17.56	8 Jan 1997 12:16:52.87	395.310
52	8 Jan 1997 19:11:27.44	8 Jan 1997 19:20:19.94	532.500
53	8 Jan 1997 20:51:05.09	8 Jan 1997 21:03:07.23	722.140
54	8 Jan 1997 22:32:28.62	8 Jan 1997 22:42:27.23	598.610
55	9 Jan 1997 08:13:13.28	9 Jan 1997 08:23:06.97	593.688
56	9 Jan 1997 09:52:29.79	9 Jan 1997 10:04:32.34	722.555
57	9 Jan 1997 11:35:13.53	9 Jan 1997 11:44:12.42	538.890
58	9 Jan 1997 18:38:49.77	9 Jan 1997 18:45:14.67	384.900
59	9 Jan 1997 20:17:26.59	9 Jan 1997 20:29:07.74	701.155
60	9 Jan 1997 21:58:20.29	9 Jan 1997 22:09:29.14	668.857
61	9 Jan 1997 23:42:16.16	9 Jan 1997 23:46:01.26	225.107

Global Statistics

Min Duration	14	3 Jan 1997 00:18:00.84	3 Jan 1997 00:20:17.27	136.427
Max Duration	6	1 Jan 1997 21:25:51.98	1 Jan 1997 21:37:55.35	723.372
Mean Duration			557.981	
Total Duration			34036.837	

Latitude -55 degrees Elevation 20 degrees
11 Apr 1998 14:09:55
Chain-Sat_ground_lat_neg55: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 07:44:00.95	1 Jan 1997 07:47:45.39	224.434
2	1 Jan 1997 09:20:48.79	1 Jan 1997 09:31:25.31	636.523
3	1 Jan 1997 11:01:21.57	1 Jan 1997 11:12:10.94	649.377
4	1 Jan 1997 12:45:27.55	1 Jan 1997 12:50:01.00	273.451
5	1 Jan 1997 19:46:58.32	1 Jan 1997 19:55:23.50	505.179
6	1 Jan 1997 21:26:34.21	1 Jan 1997 21:37:55.35	681.143
7	1 Jan 1997 23:08:08.22	1 Jan 1997 23:16:60.00	531.776
8	2 Jan 1997 08:47:47.01	2 Jan 1997 08:57:17.90	570.886
9	2 Jan 1997 10:27:18.72	2 Jan 1997 10:38:36.10	677.385
10	2 Jan 1997 12:10:18.05	2 Jan 1997 12:17:49.93	451.880
11	2 Jan 1997 19:14:20.52	2 Jan 1997 19:20:23.58	363.052
12	2 Jan 1997 20:52:55.34	2 Jan 1997 21:04:00.19	664.850
13	2 Jan 1997 22:33:54.65	2 Jan 1997 22:44:06.80	612.149
14	3 Jan 1997 00:18:51.97	3 Jan 1997 00:20:17.27	85.301
15	3 Jan 1997 08:15:05.79	3 Jan 1997 08:22:53.99	468.203
16	3 Jan 1997 09:53:33.61	3 Jan 1997 10:04:52.23	678.623
17	3 Jan 1997 11:35:30.10	3 Jan 1997 11:44:51.77	561.666
18	3 Jan 1997 18:43:58.37	3 Jan 1997 18:45:00.60	62.235
19	3 Jan 1997 20:19:26.38	3 Jan 1997 20:29:47.07	620.686
20	3 Jan 1997 21:59:53.98	3 Jan 1997 22:10:54.21	660.236
21	3 Jan 1997 23:42:39.72	3 Jan 1997 23:48:25.85	346.125
22	4 Jan 1997 07:42:48.76	4 Jan 1997 07:47:58.65	309.890
23	4 Jan 1997 09:20:06.25	4 Jan 1997 09:30:59.57	653.322
24	4 Jan 1997 11:01:01.89	4 Jan 1997 11:11:33.27	631.383
25	4 Jan 1997 12:45:34.54	4 Jan 1997 12:48:15.47	160.928
26	4 Jan 1997 19:46:10.91	4 Jan 1997 19:55:15.21	544.296
27	4 Jan 1997 21:26:03.04	4 Jan 1997 21:37:23.45	680.410
28	4 Jan 1997 23:07:52.61	4 Jan 1997 23:16:03.78	491.170
29	5 Jan 1997 08:46:57.23	5 Jan 1997 08:56:56.78	599.548
30	5 Jan 1997 10:26:52.23	5 Jan 1997 10:38:02.25	670.020
31	5 Jan 1997 12:10:16.21	5 Jan 1997 12:16:51.87	395.661
32	5 Jan 1997 19:13:18.45	5 Jan 1997 19:20:23.33	424.876
33	5 Jan 1997 20:52:20.82	5 Jan 1997 21:03:34.98	674.160
34	5 Jan 1997 22:33:32.52	5 Jan 1997 22:43:18.44	585.923
35	6 Jan 1997 08:14:07.94	6 Jan 1997 08:22:40.09	512.145
36	6 Jan 1997 09:53:00.44	6 Jan 1997 10:04:21.68	681.243
37	6 Jan 1997 11:35:20.36	6 Jan 1997 11:44:06.02	525.659
38	6 Jan 1997 18:41:26.24	6 Jan 1997 18:45:09.44	223.195

39	6 Jan 1997 20:18:47.86	6 Jan 1997 20:29:28.72	640.858
40	6 Jan 1997 21:59:27.51	6 Jan 1997 22:10:13.00	645.490
41	6 Jan 1997 23:42:45.99	6 Jan 1997 23:47:16.04	270.049
42	7 Jan 1997 07:41:41.14	7 Jan 1997 07:47:59.75	378.616
43	7 Jan 1997 09:19:26.33	7 Jan 1997 09:30:32.41	666.080
44	7 Jan 1997 11:00:44.86	7 Jan 1997 11:10:53.85	608.991
45	7 Jan 1997 19:45:26.63	7 Jan 1997 19:55:04.06	577.438
46	7 Jan 1997 21:25:33.10	7 Jan 1997 21:36:49.02	675.919
47	7 Jan 1997 23:07:40.63	7 Jan 1997 23:15:04.27	443.641
48	8 Jan 1997 07:09:42.96	8 Jan 1997 07:12:17.58	154.619
49	8 Jan 1997 08:46:10.26	8 Jan 1997 08:56:33.73	623.464
50	8 Jan 1997 10:26:28.29	8 Jan 1997 10:37:27.05	658.760
51	8 Jan 1997 12:10:17.56	8 Jan 1997 12:15:44.74	327.180
52	8 Jan 1997 19:12:23.60	8 Jan 1997 19:20:19.94	476.343
53	8 Jan 1997 20:51:47.61	8 Jan 1997 21:03:07.23	679.625
54	8 Jan 1997 22:33:12.49	8 Jan 1997 22:42:27.23	554.740
55	9 Jan 1997 08:13:13.28	9 Jan 1997 08:22:22.98	549.692
56	9 Jan 1997 09:52:29.79	9 Jan 1997 10:03:49.90	680.111
57	9 Jan 1997 11:35:13.53	9 Jan 1997 11:43:16.74	483.216
58	9 Jan 1997 18:39:58.97	9 Jan 1997 18:45:14.67	315.695
59	9 Jan 1997 20:18:10.98	9 Jan 1997 20:29:07.74	656.762
60	9 Jan 1997 21:59:02.53	9 Jan 1997 22:09:29.14	626.612
61	9 Jan 1997 23:43:09.18	9 Jan 1997 23:46:01.26	172.083

Global Statistics

Min Duration	18	3 Jan 1997 18:43:58.37	3 Jan 1997 18:45:00.60	62.235
Max Duration	36	6 Jan 1997 09:53:00.44	6 Jan 1997 10:04:21.68	681.243
Mean Duration			507.459	
Total Duration			30954.972	

Latitude 55 degrees Elevation 10 degrees
11 Apr 1998 14:07:03
Chain-Sat_ground_lat55: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 00:10:59.10	1 Jan 1997 00:22:36.64	697.540
2	1 Jan 1997 01:50:45.55	1 Jan 1997 02:03:44.01	778.466
3	1 Jan 1997 03:32:37.08	1 Jan 1997 03:44:22.93	705.855
4	1 Jan 1997 05:18:27.12	1 Jan 1997 05:24:05.90	338.775
5	1 Jan 1997 10:29:30.31	1 Jan 1997 10:40:34.82	664.515
6	1 Jan 1997 12:10:01.86	1 Jan 1997 12:22:55.94	774.077
7	1 Jan 1997 13:51:00.83	1 Jan 1997 14:03:14.41	733.587
8	1 Jan 1997 15:32:56.75	1 Jan 1997 15:41:25.21	508.453
9	1 Jan 1997 23:37:55.68	1 Jan 1997 23:48:35.16	639.484
10	2 Jan 1997 01:17:04.23	2 Jan 1997 01:29:55.58	771.350
11	2 Jan 1997 02:58:19.59	2 Jan 1997 03:10:42.21	742.615
12	2 Jan 1997 04:42:10.77	2 Jan 1997 04:50:53.84	523.069
13	2 Jan 1997 09:55:57.31	2 Jan 1997 10:05:54.89	597.580
14	2 Jan 1997 11:36:17.47	2 Jan 1997 11:48:54.37	756.898
15	2 Jan 1997 13:17:08.15	2 Jan 1997 13:29:50.21	762.055
16	2 Jan 1997 14:58:37.83	2 Jan 1997 15:08:38.71	600.881
17	2 Jan 1997 23:05:03.38	2 Jan 1997 23:14:24.40	561.019
18	3 Jan 1997 00:43:34.08	3 Jan 1997 00:56:04.55	750.465
19	3 Jan 1997 02:24:12.39	3 Jan 1997 02:36:58.98	766.592
20	3 Jan 1997 04:06:51.19	3 Jan 1997 04:17:25.05	633.857
21	3 Jan 1997 09:22:34.18	3 Jan 1997 09:29:59.58	445.404
22	3 Jan 1997 11:02:35.36	3 Jan 1997 11:14:42.26	726.900
23	3 Jan 1997 12:43:18.41	3 Jan 1997 12:56:14.79	776.380
24	3 Jan 1997 14:24:32.12	3 Jan 1997 14:35:41.19	669.071
25	3 Jan 1997 16:08:33.40	3 Jan 1997 16:10:27.53	114.127
26	3 Jan 1997 22:32:36.00	3 Jan 1997 22:39:54.03	438.037
27	4 Jan 1997 00:10:15.17	4 Jan 1997 00:22:10.17	715.006
28	4 Jan 1997 01:50:15.90	4 Jan 1997 02:03:13.58	777.679
29	4 Jan 1997 03:32:20.77	4 Jan 1997 03:43:49.38	688.607
30	4 Jan 1997 05:19:10.92	4 Jan 1997 05:23:09.37	238.451
31	4 Jan 1997 08:49:44.65	4 Jan 1997 08:53:12.07	207.418
32	4 Jan 1997 10:28:56.06	4 Jan 1997 10:40:20.15	684.096
33	4 Jan 1997 12:09:31.05	4 Jan 1997 12:22:28.25	777.199
34	4 Jan 1997 13:50:33.55	4 Jan 1997 14:02:32.48	718.936
35	4 Jan 1997 15:32:45.04	4 Jan 1997 15:40:22.74	457.703
36	4 Jan 1997 23:37:07.50	4 Jan 1997 23:48:11.08	663.577
37	5 Jan 1997 01:16:30.39	5 Jan 1997 01:29:26.02	775.628
38	5 Jan 1997 02:57:59.56	5 Jan 1997 03:10:09.78	730.217

39	5 Jan 1997 04:42:31.60	5 Jan 1997 04:50:13.32	461.721
40	5 Jan 1997 09:55:20.85	5 Jan 1997 10:05:48.93	628.080
41	5 Jan 1997 11:35:45.85	5 Jan 1997 11:48:30.76	764.909
42	5 Jan 1997 13:16:39.53	5 Jan 1997 13:29:12.53	752.998
43	5 Jan 1997 14:58:17.40	5 Jan 1997 15:07:46.73	569.323
44	5 Jan 1997 23:04:11.02	5 Jan 1997 23:14:04.49	593.477
45	6 Jan 1997 00:42:56.01	6 Jan 1997 00:55:36.03	760.024
46	6 Jan 1997 02:23:48.41	6 Jan 1997 02:36:27.44	759.031
47	6 Jan 1997 04:06:39.38	6 Jan 1997 04:16:48.86	609.481
48	6 Jan 1997 09:21:52.99	6 Jan 1997 09:30:21.73	508.739
49	6 Jan 1997 11:02:02.84	6 Jan 1997 11:14:22.58	739.746
50	6 Jan 1997 12:42:48.76	6 Jan 1997 12:55:41.34	772.579
51	6 Jan 1997 14:24:07.76	6 Jan 1997 14:34:53.40	645.642
52	6 Jan 1997 22:31:25.52	6 Jan 1997 22:39:43.72	498.200
53	7 Jan 1997 00:09:32.84	7 Jan 1997 00:21:43.04	730.199
54	7 Jan 1997 01:49:47.83	7 Jan 1997 02:02:42.84	775.015
55	7 Jan 1997 03:32:05.78	7 Jan 1997 03:43:15.27	669.490
56	7 Jan 1997 05:20:45.52	7 Jan 1997 05:21:24.51	38.990
57	7 Jan 1997 08:48:43.88	7 Jan 1997 08:54:00.72	316.843
58	7 Jan 1997 10:28:22.37	7 Jan 1997 10:40:04.17	701.799
59	7 Jan 1997 12:09:00.54	7 Jan 1997 12:21:58.99	778.448
60	7 Jan 1997 13:50:06.91	7 Jan 1997 14:01:48.94	702.036
61	7 Jan 1997 15:32:40.51	7 Jan 1997 15:38:50.29	369.780
62	7 Jan 1997 22:00:15.89	7 Jan 1997 22:04:41.11	265.222
63	7 Jan 1997 23:36:20.92	7 Jan 1997 23:47:45.96	685.032
64	8 Jan 1997 01:15:58.14	8 Jan 1997 01:28:56.11	777.970
65	8 Jan 1997 02:57:40.97	8 Jan 1997 03:09:36.96	715.993
66	8 Jan 1997 04:42:58.36	8 Jan 1997 04:49:29.54	391.178
67	8 Jan 1997 09:54:45.39	8 Jan 1997 10:05:36.27	650.879
68	8 Jan 1997 11:35:14.54	8 Jan 1997 11:48:05.62	771.080
69	8 Jan 1997 13:16:11.37	8 Jan 1997 13:28:33.24	741.875
70	8 Jan 1997 14:57:59.36	8 Jan 1997 15:06:53.18	533.827
71	8 Jan 1997 23:03:20.22	8 Jan 1997 23:13:42.68	622.455
72	9 Jan 1997 00:42:19.54	9 Jan 1997 00:55:07.09	767.553
73	9 Jan 1997 02:23:25.94	9 Jan 1997 02:35:55.57	749.637
74	9 Jan 1997 04:06:52.61	9 Jan 1997 04:16:11.45	558.841
75	9 Jan 1997 09:21:14.13	9 Jan 1997 09:30:38.84	564.715
76	9 Jan 1997 11:01:30.67	9 Jan 1997 11:14:01.44	750.765
77	9 Jan 1997 12:42:19.47	9 Jan 1997 12:55:06.29	766.820
78	9 Jan 1997 14:23:44.62	9 Jan 1997 14:34:04.02	619.408
79	9 Jan 1997 22:30:30.61	9 Jan 1997 22:39:28.58	537.971

Min Duration	56	7 Jan 1997 05:20:45.52	7 Jan 1997 05:21:24.51	38.990
Max Duration	2	1 Jan 1997 01:50:45.55	1 Jan 1997 02:03:44.01	778.466
Mean Duration			626.954	
Total Duration			49529.341	

Latitude 55 degrees Elevation 15 degrees
11 Apr 1998 14:02:08
Chain-Sat_ground_lat55: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 00:11:38.73	1 Jan 1997 00:22:36.64	657.915
2	1 Jan 1997 01:51:36.07	1 Jan 1997 02:03:44.01	727.942
3	1 Jan 1997 03:33:03.38	1 Jan 1997 03:44:22.93	679.556
4	1 Jan 1997 05:18:27.12	1 Jan 1997 05:24:05.90	338.775
5	1 Jan 1997 10:29:30.31	1 Jan 1997 10:40:20.12	649.815
6	1 Jan 1997 12:10:01.86	1 Jan 1997 12:22:08.21	726.350
7	1 Jan 1997 13:51:00.83	1 Jan 1997 14:02:28.62	687.796
8	1 Jan 1997 15:32:56.75	1 Jan 1997 15:41:19.53	502.779
9	1 Jan 1997 23:38:24.82	1 Jan 1997 23:48:35.16	610.343
10	2 Jan 1997 01:17:55.24	2 Jan 1997 01:29:55.58	720.335
11	2 Jan 1997 02:58:56.73	2 Jan 1997 03:10:42.21	705.481
12	2 Jan 1997 04:42:10.77	2 Jan 1997 04:50:53.84	523.069
13	2 Jan 1997 09:55:57.31	2 Jan 1997 10:05:54.89	597.580
14	2 Jan 1997 11:36:17.47	2 Jan 1997 11:48:12.75	715.283
15	2 Jan 1997 13:17:08.15	2 Jan 1997 13:29:00.17	712.019
16	2 Jan 1997 14:58:37.83	2 Jan 1997 15:08:16.62	578.795
17	2 Jan 1997 23:05:18.26	2 Jan 1997 23:14:24.40	546.142
18	3 Jan 1997 00:44:22.52	3 Jan 1997 00:56:04.55	702.032
19	3 Jan 1997 02:24:57.26	3 Jan 1997 02:36:58.98	721.717
20	3 Jan 1997 04:06:57.59	3 Jan 1997 04:17:25.05	627.455
21	3 Jan 1997 09:22:34.18	3 Jan 1997 09:29:59.58	445.404
22	3 Jan 1997 11:02:35.36	3 Jan 1997 11:14:09.84	694.482
23	3 Jan 1997 12:43:18.41	3 Jan 1997 12:55:23.63	725.217
24	3 Jan 1997 14:24:32.12	3 Jan 1997 14:35:06.66	634.544
25	3 Jan 1997 16:08:33.40	3 Jan 1997 16:10:27.53	114.127
26	3 Jan 1997 22:32:36.00	3 Jan 1997 22:39:54.03	438.037
27	4 Jan 1997 00:10:57.84	4 Jan 1997 00:22:10.17	672.335
28	4 Jan 1997 01:51:05.45	4 Jan 1997 02:03:13.58	728.130
29	4 Jan 1997 03:32:42.17	4 Jan 1997 03:43:49.38	667.207
30	4 Jan 1997 05:19:10.92	4 Jan 1997 05:23:09.37	238.451
31	4 Jan 1997 08:49:44.65	4 Jan 1997 08:53:12.07	207.418
32	4 Jan 1997 10:28:56.06	4 Jan 1997 10:40:00.02	663.962
33	4 Jan 1997 12:09:31.05	4 Jan 1997 12:21:39.01	727.956
34	4 Jan 1997 13:50:33.55	4 Jan 1997 14:01:49.14	675.593
35	4 Jan 1997 15:32:45.04	4 Jan 1997 15:40:22.74	457.703
36	4 Jan 1997 23:37:41.03	4 Jan 1997 23:48:11.08	630.046
37	5 Jan 1997 01:17:21.58	5 Jan 1997 01:29:26.02	724.441
38	5 Jan 1997 02:58:32.96	5 Jan 1997 03:10:09.78	696.816

39	5 Jan 1997 04:42:31.60	5 Jan 1997 04:50:13.32	461.721
40	5 Jan 1997 09:55:20.85	5 Jan 1997 10:05:44.07	623.212
41	5 Jan 1997 11:35:45.85	5 Jan 1997 11:47:46.47	720.622
42	5 Jan 1997 13:16:39.53	5 Jan 1997 13:28:23.72	704.194
43	5 Jan 1997 14:58:17.40	5 Jan 1997 15:07:30.36	552.956
44	5 Jan 1997 23:04:31.75	5 Jan 1997 23:14:04.49	572.740
45	6 Jan 1997 00:43:45.78	6 Jan 1997 00:55:36.03	710.248
46	6 Jan 1997 02:24:30.72	6 Jan 1997 02:36:27.44	716.721
47	6 Jan 1997 04:06:39.38	6 Jan 1997 04:16:48.86	609.478
48	6 Jan 1997 09:21:52.99	6 Jan 1997 09:30:21.73	508.739
49	6 Jan 1997 11:02:02.84	6 Jan 1997 11:13:46.32	703.485
50	6 Jan 1997 12:42:48.76	6 Jan 1997 12:54:50.24	721.482
51	6 Jan 1997 14:24:07.76	6 Jan 1997 14:34:23.14	615.376
52	6 Jan 1997 22:31:29.46	6 Jan 1997 22:39:43.72	494.262
53	7 Jan 1997 00:10:18.08	7 Jan 1997 00:21:43.04	684.965
54	7 Jan 1997 01:50:35.97	7 Jan 1997 02:02:42.84	726.875
55	7 Jan 1997 03:32:21.85	7 Jan 1997 03:43:15.27	653.420
56	7 Jan 1997 05:20:45.52	7 Jan 1997 05:21:24.51	38.990
57	7 Jan 1997 08:48:43.88	7 Jan 1997 08:54:00.72	316.843
58	7 Jan 1997 10:28:22.37	7 Jan 1997 10:39:39.03	676.660
59	7 Jan 1997 12:09:00.54	7 Jan 1997 12:21:08.67	728.123
60	7 Jan 1997 13:50:06.91	7 Jan 1997 14:01:08.52	661.619
61	7 Jan 1997 15:32:40.51	7 Jan 1997 15:38:50.29	369.780
62	7 Jan 1997 22:00:15.89	7 Jan 1997 22:04:41.11	265.222
63	7 Jan 1997 23:36:58.33	7 Jan 1997 23:47:45.96	647.631
64	8 Jan 1997 01:16:49.07	8 Jan 1997 01:28:56.11	727.047
65	8 Jan 1997 02:58:10.19	8 Jan 1997 03:09:36.96	686.768
66	8 Jan 1997 04:42:58.36	8 Jan 1997 04:49:29.54	391.178
67	8 Jan 1997 09:54:45.39	8 Jan 1997 10:05:25.29	639.903
68	8 Jan 1997 11:35:14.54	8 Jan 1997 11:47:19.10	724.560
69	8 Jan 1997 13:16:11.37	8 Jan 1997 13:27:46.12	694.756
70	8 Jan 1997 14:57:59.36	8 Jan 1997 15:06:43.13	523.767
71	8 Jan 1997 23:03:46.25	8 Jan 1997 23:13:42.68	596.428
72	9 Jan 1997 00:43:10.21	9 Jan 1997 00:55:07.09	716.882
73	9 Jan 1997 02:24:05.24	9 Jan 1997 02:35:55.57	710.332
74	9 Jan 1997 04:06:52.61	9 Jan 1997 04:16:11.45	558.841
75	9 Jan 1997 09:21:14.13	9 Jan 1997 09:30:38.84	564.715
76	9 Jan 1997 11:01:30.67	9 Jan 1997 11:13:21.78	711.106
77	9 Jan 1997 12:42:19.47	9 Jan 1997 12:54:15.70	716.226
78	9 Jan 1997 14:23:44.62	9 Jan 1997 14:33:38.56	593.938
79	9 Jan 1997 22:30:41.39	9 Jan 1997 22:39:28.58	527.184

Global Statistics

Min Duration	56	7 Jan 1997 05:20:45.52	7 Jan 1997 05:21:24.51	38.990
Max Duration	28	4 Jan 1997 01:51:05.45	4 Jan 1997 02:03:13.58	728.130
Mean Duration			600.532	
Total Duration			47442.048	

Global Statistics

Latitude 55 degrees Elevation 20 degrees
11 Apr 1998 14:09:26
Chain-Sat_ground_lat55: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 00:12:10.09	1 Jan 1997 00:22:36.64	626.555
2	1 Jan 1997 01:52:17.00	1 Jan 1997 02:03:44.01	687.011
3	1 Jan 1997 03:33:24.15	1 Jan 1997 03:44:22.93	658.783
4	1 Jan 1997 05:18:27.12	1 Jan 1997 05:24:05.90	338.775
5	1 Jan 1997 10:29:30.31	1 Jan 1997 10:40:08.64	638.336
6	1 Jan 1997 12:10:01.86	1 Jan 1997 12:21:29.64	687.779
7	1 Jan 1997 13:51:00.83	1 Jan 1997 14:01:51.97	651.146
8	1 Jan 1997 15:32:56.75	1 Jan 1997 15:41:15.30	498.545
9	1 Jan 1997 23:38:47.46	1 Jan 1997 23:48:35.16	587.703
10	2 Jan 1997 01:18:36.53	2 Jan 1997 01:29:55.58	679.055
11	2 Jan 1997 02:59:26.40	2 Jan 1997 03:10:42.21	675.812
12	2 Jan 1997 04:42:10.77	2 Jan 1997 04:50:53.84	523.069
13	2 Jan 1997 09:55:57.31	2 Jan 1997 10:05:54.89	597.580
14	2 Jan 1997 11:36:17.47	2 Jan 1997 11:47:39.34	681.872
15	2 Jan 1997 13:17:08.15	2 Jan 1997 13:28:19.78	671.630
16	2 Jan 1997 14:58:37.83	2 Jan 1997 15:07:59.67	561.844
17	2 Jan 1997 23:05:29.53	2 Jan 1997 23:14:24.40	534.864
18	3 Jan 1997 00:45:01.48	3 Jan 1997 00:56:04.55	663.068
19	3 Jan 1997 02:25:33.42	3 Jan 1997 02:36:58.98	685.561
20	3 Jan 1997 04:07:02.56	3 Jan 1997 04:17:25.05	622.493
21	3 Jan 1997 09:22:34.18	3 Jan 1997 09:29:59.58	445.404
22	3 Jan 1997 11:02:35.36	3 Jan 1997 11:13:44.07	668.712
23	3 Jan 1997 12:43:18.41	3 Jan 1997 12:54:42.18	683.771
24	3 Jan 1997 14:24:32.12	3 Jan 1997 14:34:39.58	607.465
25	3 Jan 1997 16:08:33.40	3 Jan 1997 16:10:27.53	114.127
26	3 Jan 1997 22:32:36.00	3 Jan 1997 22:39:54.03	438.037
27	4 Jan 1997 00:11:31.80	4 Jan 1997 00:22:10.17	638.379
28	4 Jan 1997 01:51:45.56	4 Jan 1997 02:03:13.58	688.019
29	4 Jan 1997 03:32:58.99	4 Jan 1997 03:43:49.38	650.386
30	4 Jan 1997 05:19:10.92	4 Jan 1997 05:23:09.37	238.451
31	4 Jan 1997 08:49:44.65	4 Jan 1997 08:53:12.07	207.418
32	4 Jan 1997 10:28:56.06	4 Jan 1997 10:39:44.21	648.157
33	4 Jan 1997 12:09:31.05	4 Jan 1997 12:20:59.15	688.102
34	4 Jan 1997 13:50:33.55	4 Jan 1997 14:01:14.61	641.059
35	4 Jan 1997 15:32:45.04	4 Jan 1997 15:40:22.74	457.703
36	4 Jan 1997 23:38:07.29	4 Jan 1997 23:48:11.08	603.793
37	5 Jan 1997 01:18:03.04	5 Jan 1997 01:29:26.02	682.981
38	5 Jan 1997 02:58:59.54	5 Jan 1997 03:10:09.78	670.237

39	5 Jan 1997 04:42:31.60	5 Jan 1997 04:50:13.32	461.721
40	5 Jan 1997 09:55:20.85	5 Jan 1997 10:05:40.30	619.443
41	5 Jan 1997 11:35:45.85	5 Jan 1997 11:47:10.81	684.962
42	5 Jan 1997 13:16:39.53	5 Jan 1997 13:27:44.43	664.903
43	5 Jan 1997 14:58:17.40	5 Jan 1997 15:07:17.92	540.517
44	5 Jan 1997 23:04:47.63	5 Jan 1997 23:14:04.49	556.860
45	6 Jan 1997 00:44:25.94	6 Jan 1997 00:55:36.03	670.093
46	6 Jan 1997 02:25:04.71	6 Jan 1997 02:36:27.44	682.725
47	6 Jan 1997 04:06:39.38	6 Jan 1997 04:16:48.86	609.476
48	6 Jan 1997 09:21:52.99	6 Jan 1997 09:30:21.73	508.739
49	6 Jan 1997 11:02:02.84	6 Jan 1997 11:13:17.38	674.542
50	6 Jan 1997 12:42:48.76	6 Jan 1997 12:54:08.88	680.122
51	6 Jan 1997 14:24:07.76	6 Jan 1997 14:33:59.58	591.816
52	6 Jan 1997 22:31:32.39	6 Jan 1997 22:39:43.72	491.331
53	7 Jan 1997 00:10:54.24	7 Jan 1997 00:21:43.04	648.798
54	7 Jan 1997 01:51:14.89	7 Jan 1997 02:02:42.84	687.956
55	7 Jan 1997 03:32:34.42	7 Jan 1997 03:43:15.27	640.854
56	7 Jan 1997 05:20:45.52	7 Jan 1997 05:21:24.51	38.990
57	7 Jan 1997 08:48:43.88	7 Jan 1997 08:54:00.72	316.843
58	7 Jan 1997 10:28:22.37	7 Jan 1997 10:39:19.20	656.827
59	7 Jan 1997 12:09:00.54	7 Jan 1997 12:20:27.90	687.358
60	7 Jan 1997 13:50:06.91	7 Jan 1997 14:00:36.49	629.588
61	7 Jan 1997 15:32:40.51	7 Jan 1997 15:38:50.29	369.780
62	7 Jan 1997 22:00:15.89	7 Jan 1997 22:04:41.11	265.222
63	7 Jan 1997 23:37:27.81	7 Jan 1997 23:47:45.96	618.149
64	8 Jan 1997 01:17:30.33	8 Jan 1997 01:28:56.11	685.786
65	8 Jan 1997 02:58:33.35	8 Jan 1997 03:09:36.96	663.615
66	8 Jan 1997 04:42:58.36	8 Jan 1997 04:49:29.54	391.178
67	8 Jan 1997 09:54:45.39	8 Jan 1997 10:05:16.75	631.360
68	8 Jan 1997 11:35:14.54	8 Jan 1997 11:46:41.55	687.012
69	8 Jan 1997 13:16:11.37	8 Jan 1997 13:27:08.31	656.946
70	8 Jan 1997 14:57:59.36	8 Jan 1997 15:06:35.56	516.203
71	8 Jan 1997 23:04:06.36	8 Jan 1997 23:13:42.68	576.316
72	9 Jan 1997 00:43:51.17	9 Jan 1997 00:55:07.09	675.919
73	9 Jan 1997 02:24:36.72	9 Jan 1997 02:35:55.57	678.855
74	9 Jan 1997 04:06:52.61	9 Jan 1997 04:16:11.45	558.841
75	9 Jan 1997 09:21:14.13	9 Jan 1997 09:30:38.84	564.715
76	9 Jan 1997 11:01:30.67	9 Jan 1997 11:12:50.01	679.333
77	9 Jan 1997 12:42:19.47	9 Jan 1997 12:53:34.81	675.333
78	9 Jan 1997 14:23:44.62	9 Jan 1997 14:33:18.89	574.276
79	9 Jan 1997 22:30:49.51	9 Jan 1997 22:39:28.58	519.063

Global Statistics

Min Duration	56	7 Jan 1997 05:20:45.52	7 Jan 1997 05:21:24.51	38.990
Max Duration	33	4 Jan 1997 12:09:31.05	4 Jan 1997 12:20:59.15	688.102
Mean Duration			579.444	
Total Duration			45776.048	

Appendix 2

11 Apr 1998 16:20:31
Chain-Chain1: Complete Chain Access

Access	Start Time (UTCG)	Stop Time (UTCG)	Duration (sec)
1	1 Jan 1997 00:04:55.51	1 Jan 1997 00:25:06.78	1211.274

Appendix 3

```
/*pre-processor library links*/
#include <stdio.h>
#include <math.h>
#include <values.h>
#include <iostream.h>
/*various user defined structures */
struct adj_matrix
 {int A[66][66];
 };

struct sat_info /*if tag=0 then satellite has not been reached by a g_s*/
 {int tag;  /* if tag=1 then satellite has been reached by a ground satition*/
  float lon;  /*longitude of sub-satellite point at initial geometry, time 0*/
  float lat;  /*latitude of sub-satellite point at initial geometry, time 0*/
 };

struct all_sats
 { sat_info x[66];
 };

struct gstations_info
 { int z; /* number of ground stations needed*/
        float lat[66]; /*latitude of each ground station*/
        float lon[66]; /*longitude of each groudn station*/
        int   check;
};


/*funciton headers */
adj_matrix Adj_matrix(int,int);
adj_matrix Adj_matrix_mid(int,int);
FILE* fopener(void);
int get_sat(int,int,all_sats);
int shortest_path(adj_matrix,  int , int );
gstations_info how_many_gs(float,float,all_sats,int,int,int,int);
int test_gs(gstations_info,float,float,all_sats, int , int ,int ,int);


/*Global Variables*/
FILE *latp;
FILE *lonp;
FILE *testp;
adj_matrix pre_ep,post_ep,mid_ep;
```

```
main()
 { int planes,sats_per_plane,i,gpt,clpt,spt,count,p,p2,check;
        float epsilon,mean_tiv;
        all_sats iridium;
        gstations_info optimal,constrained;

        cout<<"open test file"<<endl;
        testp=fopener();
        /*Declare variables for the iridium system*/
        clpt=2; /*command load processing time at ground station*/
        spt=1; /*satellite processing time*/
        gpt=5; /*ground station health and safety data processing time*/
        planes=6;
        sats_per_plane=11;
        epsilon=207.0; /*time before intial geometry changes*/
        mean_tiv=572.7903333; /*mean time in view */

        /*compute the adjancey matrices before and after epsilon*/
        cout<<"Enter data on pre_epsilon crosslinks"<<endl;
        pre_ep=Adj_matrix(planes,sats_per_plane);
        cout<<"Enter data on post_epsilon crosslinks"<<endl;
        post_ep=Adj_matrix(planes,sats_per_plane);
        cout<<"Enter data on transitional crosslinks"<<endl;
        mid_ep=Adj_matrix_mid(planes,sats_per_plane);
        /* set the tags for the iridium structures equal to all zero indicating
        that the satellite has yet to be commanded by a ground station*/
        /* initialize the longitude and latitude for the satellites to their
                sub-satellite point longitude and latitude at initial geometry, ie time
                zero*/
        cout<<endl;
        cout<<"Enter the name of the file which has sat's latitude info"<<endl;
        latp=fopener();
        cout<<endl;
        cout<<"Enter the name of the file which has sat's longitude info"<<endl;
        lonp=fopener();
        cout<<endl;
        for(i=0;i<=65;i++)
         {iridium.x[i].tag=0;
          fscanf(latp,"%f",&iridium.x[i].lat);
          fscanf(lonp,"%f",&iridium.x[i].lon);
          }

/*call how_many_gs to determine the optimal number without regard to land or
political constraints*/
/*      for(p=1;p<=82;p=p+9) /*p is the range of delay times*/
```

```
/*      {cout<<"for this run p="<<p<<endl;
            optimal=how_many_gs(mean_tiv,epsilon,iridium,p,gpt,clpt,spt);
            if(optimal.z>=65)
                  p=MAXINT;
            } */
/*use test_gs to test different ground stations with a given value of p to
adjust the optimal set to account for land and political constraints*/
/*Note that you must set up the gstations_info, constrained, each time you
want to check a new set of ground stations, you must also modify p2*/
 p2=1;
 constrained.z=55;
 constrained.lat[0]=40; constrained.lon[0]=-80;
 constrained.lat[1]=-15; constrained.lon[1]=-40;
 check=test_gs(constrained,mean_tiv,epsilon,iridium,p2,gpt,clpt,spt);
 cout<<"check="<<check<<endl;
      return(0);
 }

/* The subroutine how_many_gs is the main subroutine.  It is basically a giant
loop that given a value of p, the delay time on a satellite, computes the
number of ground stations that will be required to command Iridium within one
satellite pass.  It takes several pieces of information which are:
double Mean time in view, double epsilon- the time at which geometry changes,
two adjancey matrices one for pre-epsilon time and one for post-epsilon time,
an all_sats which contains the longitude and latitude and the tag for all of
the satellites at the begining of the pass, p the satellite delay time, and x
the processing time on the ground.  It returns a data structure of the type
g_stations_info which has the number of ground stations required for a given p
value and the longitude and latitude of each of those ground stations.  The
subroutine works by searching through each longitude and latitude position
(with  5 degree increments, although this could be altered to any desired
increment by changing the for loop, and determing how many satellites that site
will be able to command, picking the one that commands the most, then running
the loop again until all satellites have been commanded*/
gstations_info how_many_gs(float tiv,float ep, all_sats z, int p, int gpt,int clpt,int spt)
/*start how_many_gs----------------------------------------------------*/
{ all_sats temp_sats, mid_sats;
  int gs_count, watch, root_sat,sats_covered, max_sats, i,j,k, alpha,beta;
  int m;
  float request,respond,delay;
  gstations_info ansr;

  temp_sats=z; gs_count=0;
  do
  {max_sats=0;
```

```
for(i=65;i>=-65;i=i-5) /*latitude incrementor*/
 {for(j=180;j>=-175;j=j-5) /*longitude incrementor*/
        {
        for(m=0;m<=65;m++)
                        temp_sats.x[m].tag=0;
        root_sat=get_sat(i,j,z); /*find the sat closest to this lat/lon*/
        temp_sats.x[root_sat].tag=1;
        sats_covered=1;
        for(k=0;k<=65;k++)
                {if(z.x[k].tag==0 && k!=root_sat)
                        {alpha=shortest_path(pre_ep,root_sat,k);
                        request=(alpha*p);
                        if(request>ep/2.0)
                                alpha=shortest_path(mid_ep,root_sat,k);
                        request=(alpha+1)*p+spt;
                        if(request<=(ep/2.0))
                                beta=alpha;
                        if(request>(ep/2.0) && request<ep)
                                beta=shortest_path(mid_ep,k,root_sat);
                        if(request>=ep)
                                beta=shortest_path(post_ep,k,root_sat);
                        respond=(beta+1)*p;
                        delay=respond+request+gpt+clpt;
                        if(delay<tiv)
                                {temp_sats.x[k].tag=1;
                                 sats_covered=sats_covered+1;
                                }
                        }/*closes first if*/
         }/*closes k loop */
        if(sats_covered>max_sats)
                {max_sats=sats_covered;
                 ansr.lon[gs_count]=j;
                 ansr.lat[gs_count]=i;
                 mid_sats=temp_sats;
                }
        }/*closes j*/
 }/*closes i*/
for(m=0;m<=65;m++)
 { if(mid_sats.x[m].tag==1)
         z.x[m].tag=1;}
for(m=0;m<=65;m++)
 {if(z.x[m].tag==1)
        watch=0;
 }
for(m=0;m<=65;m++)
```

```
        {if(z.x[m].tag==0)
                watch=1;
        }
       gs_count=gs_count+1;
       cout<<"gscount="<<gs_count<<endl;
       cout<<"ansr.lon="<<ansr.lon[gs_count-1]<<"--ansr.lat="<<ansr.lat[gs_count-1]<<endl;
 }while(watch==1 && gs_count<=66 );
 ansr.z=gs_count; ansr.check=watch;
 fprintf(testp,"\n For p=%d, the number of ground stations was %d",p,gs_count);
 for(m=0;m<=(gs_count-1);m++)
                fprintf(testp,"\n Lat=%f \t Lon=%f \n",ansr.lat[m],ansr.lon[m]);
 return(ansr);
}/*end how_many_gs*/

/*The test_gs is used to test a set of ground stations to see if they will
satisfy the commanding requirements.  When called the function takes a data
structure of the type gstations_info, the mean time in view, epsilon, gpt,clpt,
spt,p delay time, and an all sats.  It returns an integer, check.  If check
equals 0 then the given set of ground station coordinates will successfully
command all of the ground stations.  If check equals 1 then the given set
fails.*/
int test_gs(gstations_info x,float tiv,float ep,all_sats z,int p,int gpt,int clpt,int spt)
/*start how_many_gs----------------------------------------------------*/
{
 int  root_sat, i,j,k, alpha,beta;
 int m,check,count;
 float request,respond,delay;

        for(count=0;count<=x.z-1;count++)
                { i=x.lat[count]; j=x.lon[count];
                 root_sat=get_sat(i,j,z); /*find the sat closest to this lat/lon*/
                 z.x[root_sat].tag=1;
                 for(k=0;k<=65;k++)
                        {if(z.x[k].tag==0 && k!=root_sat)
                                {alpha=shortest_path(pre_ep,root_sat,k);
                                 request=(alpha*p);
                                 if(request>ep/2.0)
                                        alpha=shortest_path(mid_ep,root_sat,k);
                                 request=(alpha+1)*p+spt;
                                 if(request<=(ep/2.0))
                                        beta=alpha;
                                 if(request>(ep/2.0) && request<ep)
                                        beta=shortest_path(mid_ep,k,root_sat);
                                 if(request>=ep)
                                        beta=shortest_path(post_ep,k,root_sat);
```

```
                                respond=(beta+1)*p;
                                delay=respond+request+gpt+clpt;
                                if(delay<tiv)
                                        {z.x[k].tag=1;
                                        }
                               }/*closes first if*/
                     }/*closes k loop */
                   }/*closes first for loop,count*/

          check=0;
          for(m=0;m<=65;m++)
            {if(z.x[m].tag==0)
                    check=1;
            }
 return(check);
}/*end how_many_gs*/

/*The shortest_path subroutine is used to find the shortest path between
 satellites A and B.  It is dependent on the adjancey matrix that is used
 when the function is callled.  The adjancey matrix is dependent on the
 current satellite geometry, that is on the cross links*/
int shortest_path(adj_matrix A, int b, int final)
{ int v1[66],v2[66],v3[66],count,max,min_path,c2,root;
  int i,wi,wj,j,k,m,max_wt,bell,wt,ansr,no_path,x;
  int e2[264][2];
  max=67; /*set max equal number of satellites plus 1*/
  /*initialize*/
  i=b;
  wi=0;
  wj=0;
  count=0;
  for(j=0;j<=1;j++)
          {for(k=0;k<=263;k++)
            {
                    e2[k][j]=max;
            }
          }
 for(j=0;j<=65;j++)
          {if(j==b)
            {v3[j]=max;
                    v2[j]=max;
                    v1[j]=b;
            }
          else
           {v3[j]=j;
```

```
        v2[j]=max;
        v1[j]=max;
       }
      }
do
 {for(j=0;j<=65;j++)
      {if(A.A[i][j]==1)
            {
            wj=1;
            if(v2[j]!=max)
                  {root=v2[j]; min_path=0;
                  for(c2=0;c2<=263;c2++)
                   {
                        if(e2[c2][0]==b && e2[c2][1]==root)
                         {min_path=min_path+1;
                              c2=265;
                         }
                        if(e2[c2][0]!=b && e2[c2][1]==root)
                         {root=e2[c2][0];
                              min_path=min_path+1;
                              c2=-1;
                         }
                        } /*remove the edge from e2 */
                  if((wi+wj)<min_path)
                  { for(c2=0;c2<=263;c2++)
                              {if(e2[c2][1]==j)
                               {e2[c2][0]=max;
                                    e2[c2][1]=max;
                               }
                               }


                        m=0;
                        do    /*add the edge to e2*/
                         {if(e2[m][0]==max && e2[m][1]==max)
                              {e2[m][0]=i;
                               e2[m][1]=j;
                               m=MAXINT;
                              }
                         else
                          m=m+1;
                         }while(m != MAXINT);
                        }
                  }
            if(v3[j]!=max)
```

```
                {v3[j]=max;
                 v2[j]=j;
                 k=0;
                 do
                        {if(e2[k][0]==max && e2[k][1]==max)
                                {e2[k][0]=i;
                                 e2[k][1]=j;
                                 k=MAXINT;
                                 }
                        else
                          k=k+1;
                        }while(k !=MAXINT );
                }
        }
}
        no_path=0;
        for(k=0;k<=65;k++)
                {if(v2[k]!=max)
                        no_path=1;
                }
        if(no_path==0)
                {cout<<"NO PATH FOR THESE SATELLITES"<<endl;
                 return(MAXINT);
                }

        max_wt=MAXINT;
        for(bell=0;bell<=65;bell++)
                {if(v2[bell]!=max)
                        {root=v2[bell]; wt=0;
                         for(c2=0;c2<=263;c2++)
                                {if(e2[c2][0]==b && e2[c2][1]==root)
                                        {wt=wt+1;
                                         c2=265;
                                         }
                                 if(e2[c2][0]!=b && e2[c2][1]==root)
                                         {root=e2[c2][0];
                                          wt=wt+1;
                                          c2=-1;
                                          }
                                 }
                         if(wt<max_wt)
                                {max_wt=wt;
                                 x=v2[bell];
                                 }
                        }
```

```
                    }

                v2[x]=max;
                v1[x]=x;
                i=x;
                wi=max_wt;
                count=count+1;
        }while(i!=final  && count<1000);

ansr=max_wt;
return(ansr);
}/*end shortest_path*/


/* The get_sat subroutine is used to find the root satellite for a given ground
station latitude and longitude position.  It is based on the intial geometry of
the system.  It will compare the ground station latitude and longitude to
the latitude and longitude of every satellite, finding the one that is closest
to the ground station coordinates and therefore the one that a ground station
at those coordinates would be in contact with. It takes two intergers i and j
which are the latitude and longitude repsectively of the ground station. It
returns an interger which is the index number of the iridium satellite that is
closest of those coordinates*/
int get_sat(int lat, int lon, all_sats z)
 { int i, ansr;
        float sat_lat, sat_lon, t1, t2, td, total_dif;

        t1=0; t2=0;  td=0; total_dif=1000;
        for(i=0;i<=65;i++)
         { sat_lat=z.x[i].lat;
                sat_lon=z.x[i].lon;
                if(fabs(lon)>=30.0 && fabs(lon)<=150 && fabs(lat)>=17)
                  {if(lon<=0.0 && lat>=0 && sat_lat>=0 && sat_lon<=0)
                        {t1=fabs(lon)-fabs(sat_lon);
                                t1=fabs(t1);
                                t2=fabs(lat-sat_lat);
                                td=t1+t2;
                                if(td<total_dif)
                                        {total_dif=td;
                                         ansr=i;
                                        }
                                }
                        if(lon>=0 && lat>=0 && sat_lat>=0 && sat_lon>=0)
                                 {t1=fabs(lon-sat_lon);
                                  t2=fabs(lat-sat_lat);
```

```
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        if(lon>=0 && lat<=0 && sat_lon>=0 && sat_lat<=0)
                {t1=fabs(lon-sat_lon);
                t2=fabs(lat)-fabs(sat_lat);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        if(lon<=0 && lat<=0 && sat_lon<=0 && sat_lat<=0)
                {t1=fabs(lon)-fabs(sat_lon);
                t1=fabs(t1);
                t2=fabs(lat)-fabs(sat_lat);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
 }
if(lat>=17)
 {if(lon>=150 && sat_lon>=120 && sat_lat>=0)
        {t1=fabs(lon-sat_lon);
                t2=fabs(lat-sat_lat);
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
        }
       if(lon>=150 && sat_lon<=-150 && sat_lat>=0)
        {t1=(180-lon)+(180-fabs(sat_lon));
                t2=fabs(lat-sat_lat);
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
```

```
                }
               }
        if(lon<=-150 && sat_lon<=-120 && sat_lat>=0)
         {t1=fabs(lon)-fabs(sat_lon);
                t1=fabs(t1);
                t2=fabs(lat-sat_lat);
                td=t1+t2;
                if(td<total_dif)
                 {total_dif=td;
                        ansr=i;
                 }
         }
        if(lon<=-150 && sat_lon>=150 && sat_lat>=0)
         {t1=(180-sat_lon)+(180-fabs(lon));
                t2=fabs(sat_lat-lat);
                td=t1+t2;
                if(td<total_dif)
                 {total_dif=td;
                        ansr=i;
                 }
         }
 }
if(lat<=-17)
 {if(lon>=150 && sat_lon>=120 && sat_lat<=0)
         {t1=fabs(lon-sat_lon);
                t2=fabs(lat)-fabs(sat_lat);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                 {total_dif=td;
                        ansr=i;
                 }
         }
        if(lon>=150 && sat_lon<=-150 && sat_lat<=0)
         {t1=(180-lon)+(180-fabs(sat_lon));
                t2=fabs(lat)-fabs(sat_lat);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                 {total_dif=td;
                        ansr=i;
                 }
               }
        if(lon<=-150 && sat_lon<=-120 && sat_lat<=0)
         {t1=fabs(lon)-fabs(sat_lon);
```

```
                t1=fabs(t1);
                t2=fabs(lat)-fabs(sat_lat);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
              }
             if(lon<=-150 && sat_lon>=150 && sat_lat<=0)
              {t1=(180-sat_lon)+(180-fabs(lon));
                t2=fabs(lat)-fabs(sat_lat);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
              }
  }
if(lat>=17 && fabs(lon)<=30 && fabs(sat_lon)<=60 && sat_lat>=0)
  {if(lon>=0 && sat_lon>=0)
          {t1=fabs(lon-sat_lon);
                t2=fabs(lat-sat_lat);
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
          }
  if(lon>=0 && sat_lon<=0)
          {t1=lon+fabs(sat_lon);
          t2=fabs(sat_lat-lat);
          td=t1+t2;
          if(td<total_dif)
                {total_dif=td;
                 ansr=i;
                }
          }
  if(lon<=0 && sat_lon>=0)
          {t1=fabs(lon) + sat_lon;
          t2=fabs(sat_lat-lat);
          td=t1+t2;
          if(td<total_dif)
                 {total_dif=td;
```

```
                    ansr=i;
                    }
                }
        if(lon<=0 && sat_lon<=0)
                {t1=fabs(lon)-fabs(sat_lon);
                t1=fabs(t1);
                t2=fabs(sat_lat-lat);
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                        ansr=i;
                        }
                }
        }

if(lat<=-17 && fabs(lon)<=30 && fabs(sat_lon)<=60 && sat_lat<=0)
        {if(lon>=0 && sat_lon>=0)
                {t1=fabs(lon-sat_lon);
                        t2=fabs(sat_lat)-fabs(lat);
                        t2=fabs(t2);
                        td=t1+t2;
                        if(td<total_dif)
                          {total_dif=td;
                                ansr=i;
                          }
                }
        if(lon>=0 && sat_lon<=0)
                {t1=lon+fabs(sat_lon);
                t2=fabs(sat_lat)-fabs(lat);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                        ansr=i;
                        }
                }
        if(lon<=0 && sat_lon>=0)
                {t1=fabs(lon) + sat_lon;
                t2=fabs(sat_lat)-fabs(lat);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                        ansr=i;
                        }
```

```
                }
        if(lon<=0 && sat_lon<=0)
                {t1=fabs(lon)-fabs(sat_lon);
                 t1=fabs(t1);
                 t2=fabs(sat_lat)-fabs(lat);
                 t2=fabs(t2);
                 td=t1+t2;
                 if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        }

if(fabs(lat)<=17)
        {if(lat>=0)
                {if(lon>=30 && lon<=150 && sat_lon>=0)
                        {if(sat_lat<=0)
                                {t1=fabs(sat_lat)+lat;
                                 t2=fabs(lon-sat_lon);
                                 td=t1+t2;
                                 if(td<total_dif)
                                        {total_dif=td;
                                         ansr=i;
                                        }
                                }
                         if(sat_lat>=0)
                          {t1=fabs(sat_lat-lat);
                                t2=fabs(lon-sat_lon);
                                td=t1+t2;
                                if(td<total_dif)
                                  {total_dif=td;
                                        ansr=i;
                                  }
                          }
                         }
                        if(lon<=-30 && lon>=-150 && sat_lon<=0)
                          {if(sat_lat<0)
                                {t1=fabs(sat_lat)+lat;
                                        t2=fabs(sat_lon)-fabs(lon);
                                        t2=fabs(t2);
                                        td=t1+t2;
                                        if(td<total_dif)
                                          {total_dif=td;
                                                ansr=i;
```

```
                }
        }
 if(sat_lat>=0)
        {t1=fabs(lat-sat_lat);
        t2=fabs(sat_lon)-fabs(lon);
        t2=fabs(t2);
        td=t1+t2;
        if(td<total_dif)
                {total_dif=td;
                ansr=i;
                }
        }
 }
if(lon>=0 && lon<=30)
 {if(sat_lat>=0 && sat_lon>=0)
        {t1=fabs(lat-sat_lat);
                t2=fabs(lon-sat_lon);
                td=t1+t2;
                if(td<total_dif)
                 {total_dif=td;
                        ansr=i;
                 }
                }
 if(sat_lat>=0 && sat_lon<=0)
        {t1=fabs(sat_lat-lat);
        t2=lon+fabs(sat_lon);
        td=t1+t2;
        if(td<total_dif)
                {total_dif=td;
                ansr=i;
                }
        }
 if(sat_lat<=0 && sat_lon>=0)
        {t1=lat+fabs(sat_lat);
        t2=fabs(sat_lon-lon);
        td=t1+t2;
        if(td<total_dif)
                {total_dif=td;
                ansr=i;
                }
        }
 if(sat_lat<=0 && sat_lon<=0)
        {t1=lat+fabs(sat_lat);
        t2=lon+fabs(sat_lon);
        td=t1+t2;
```

```
                    if(td<total_dif)
                            {total_dif=td;
                             ansr=i;
                            }
                    }
            }
if(lon<=0 && lon>=-30)
 {if(sat_lat>=0 && sat_lon>=0)
            {t1=fabs(lat-sat_lat);
             t2=fabs(lon)+sat_lon;
             td=t1+t2;
             if(td<total_dif)
                    {total_dif=td;
                     ansr=i;
                    }
             }
 if(sat_lat>=0 && sat_lon<=0)
            {t1=fabs(sat_lat-lat);
             t2=fabs(sat_lon)-fabs(lon);
             t2=fabs(t2);
             td=t1+t2;
             if(td<total_dif)
                    {total_dif=td;
                     ansr=i;
                    }
            }
 if(sat_lat<=0 && sat_lon>=0)
            {t1=lat+fabs(sat_lat);
             t2=fabs(lon)+sat_lon;
             td=t1+t2;
             if(td<total_dif)
                    {total_dif=td;
                     ansr=i;
                    }
            }
 if(sat_lat<=0 && sat_lon<=0)
            {t1=fabs(sat_lat)+lat;
             t2=fabs(lon)-fabs(sat_lon);
             t2=fabs(t2);
             td=t1+t2;
             if(td<total_dif)
                    {total_dif=td;
                     ansr=i;
                    }
            }
```

```
        }
if(lon>=150)
        {if(sat_lat>=0 && sat_lon>=0)
         {t1=fabs(lat-sat_lat);
                t2=fabs(lon-sat_lon);
                td=t1+t2;
                if(td<total_dif)
                 {total_dif=td;
                        ansr=i;
                 }
                }
        if(sat_lat>=0 && sat_lon<=-150)
         {t1=fabs(lat-sat_lat);
                t2=(180-lon)+(180-fabs(sat_lon));
                td=t1+t2;
                if(td<total_dif)
                 {total_dif=td;
                        ansr=i;
                 }
         }
        if(sat_lat<=0 && sat_lon>=0)
         {t1=fabs(lon-sat_lon);
                t2=fabs(sat_lat)+lat;
                td=t1+t2;
                if(td<total_dif)
                 {total_dif=td;
                        ansr=i;
                 }
         }
        if(sat_lat<=0 && sat_lon<=-150)
         {t1=fabs(sat_lat)+lat;
                t2=(180-lon)+(180-fabs(sat_lon));
                td=t1+t2;
                if(td<total_dif)
                 {total_dif=td;
                        ansr=i;
                 }
         }
 }
if(lon<=-150)
 {if(sat_lat>=0 && sat_lon<=0)
        {t1=fabs(lat-sat_lat);
                t2=fabs(lon)-fabs(sat_lon);
                t2=fabs(t2);
                td=t1+t2;
```

```
                        if(td<total_dif)
                          {total_dif=td;
                                ansr=i;
                          }
                }
        if(sat_lat>=0 && sat_lon>=150)
                {t1=fabs(lat-sat_lat);
                t2=(180-sat_lon)+(180-fabs(lon));
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                        ansr=i;
                        }
                }
        if(sat_lat<=0 && sat_lon<=0)
                {t2=fabs(lon)-fabs(sat_lon);
                t2=fabs(t2);
                t1=fabs(sat_lat)+lat;
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                        ansr=i;
                        }
                }
        if(sat_lat<=0 && sat_lon>=150)
                {t1=fabs(sat_lat)+lat;
                t2=(180-sat_lon)+(180-fabs(lon));
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                        ansr=i;
                        }
                }
        }
}
if(lat<=0)
        {if(lon>=30 && lon<=150 && sat_lon>=0)
                {if(sat_lat<=0)
                        {t1=fabs(lat)-fabs(sat_lat);
                        t1=fabs(t1);
                        t2=fabs(lon-sat_lon);
                        td=t1+t2;
                        if(td<total_dif)
                                {total_dif=td;
                                ansr=i;
```

```
                        }
                }
        if(sat_lat>=0)
                {t1=fabs(lat)+sat_lat;
                 t2=fabs(lon-sat_lon);
                 td=t1+t2;
                 if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
 }
if(lon<=-30 && lon>=-150 && sat_lon<=0)
        {if(sat_lat<=0)
         {t1=fabs(sat_lat)-fabs(lat);
                t1=fabs(t1);
                t2=fabs(sat_lon)-fabs(lon);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
                }
         if(sat_lat>=0)
                 {t1=fabs(lat)+sat_lat;
                  t2=fabs(sat_lon)-fabs(lon);
                  t2=fabs(t2);
                  td=t1+t2;
                  if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                 }
        }
if(lon>=0 && lon<=30)
        {if(sat_lat>=0 && sat_lon>=0)
         {t1=fabs(lat)+sat_lat;
                t2=fabs(lon-sat_lon);
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
         }
```

```
        if(sat_lat>=0 && sat_lon<=0)
          {t1=fabs(lat)+sat_lat;
                t2=lon+fabs(sat_lon);
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
          }
        if(sat_lat<=0 && sat_lon>=0)
          {t1=fabs(sat_lat)-fabs(lat);
                t1=fabs(t1);
                t2=fabs(lon-sat_lon);
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
          }
        if(sat_lat<=0 && sat_lon<=0)
          {t1=fabs(sat_lat)-fabs(lat);
                t1=fabs(t1);
                t2=fabs(sat_lon)+lon;
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
          }
 }
if(lon<=0 && lon>=-30)
        {if(sat_lat>=0 && sat_lon>=0)
          {t1=fabs(lat)+sat_lat;
                t2=fabs(lon)+sat_lon;
                td=t1+t2;
                if(td<total_dif)
                  {total_dif=td;
                        ansr=i;
                  }
          }
         if(sat_lat>=0 && sat_lon<=0)
          {t1=fabs(lat)+sat_lat;
                t2=fabs(lon)-fabs(sat_lon);
                t2=fabs(t2);
                td=t1+t2;
```

```
                        if(td<total_dif)
                          {total_dif=td;
                                ansr=i;
                          }
                  }
                 if(sat_lat<=0 && sat_lon>=0)
                  {t1=fabs(lat)-fabs(sat_lat);
                        t1=fabs(t1);
                        t2=fabs(lon)+sat_lon;
                        td=t1+t2;
                        if(td<total_dif)
                          {total_dif=td;
                                ansr=i;
                          }
                  }
                 if(sat_lat<=0  && sat_lon<=0)
                  {t1=fabs(lat)-fabs(sat_lat);
                        t1=fabs(t1);
                        t2=fabs(lon)-fabs(sat_lon);
                        t2=fabs(t2);
                        td=t1+t2;
                        if(td<total_dif)
                          {total_dif=td;
                                ansr=i;
                          }
                  }
        }
if(lon>=150)
        {if(sat_lat>=0  && sat_lon>=0)
                {t1=fabs(lat)+sat_lat;
                 t2=fabs(lon-sat_lon);
                 td=t1+t2;
                 if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        if(sat_lat>=0 && sat_lon<=-150)
                {t1=fabs(lat)+sat_lat;
                 t2=(180-lon)+(180-fabs(sat_lon));
                 td=t1+t2;
                 if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
```

```
                }
        if(sat_lat<=0  && sat_lon>=0)
                {t1=fabs(lat)-fabs(sat_lat);
                 t1=fabs(t1);
                 t2=fabs(lon-sat_lon);
                 td=t1+t2;
                 if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        if(sat_lat<=0 && sat_lon<=-150)
                {t1=fabs(lat)-fabs(sat_lat);
                 t1=fabs(t1);
                 t2=(180-lon)+(180-fabs(sat_lon));
                 td=t1+t2;
                 if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        }
if(lon<=-150)
        {if(sat_lat>=0  && sat_lon<=0)
                {t1=fabs(lat)+sat_lat;
                 t2=fabs(lon)-fabs(sat_lon);
                 t2=fabs(t2);
                 td=t1+t2;
                 if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        if(sat_lat>=0 && sat_lon>=150)
                {t1=fabs(lat)+sat_lat;
                 t2=(180-sat_lon)+(180-fabs(lon));
                 td=t1+t2;
                 if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        if(sat_lat<=0 && sat_lon<=0)
                {t1=fabs(lat)-fabs(sat_lat);
                 t1=fabs(t1);
```

```
                t2=fabs(lon)-fabs(sat_lon);
                t2=fabs(t2);
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        if(sat_lat<=0 && sat_lon>=150)
                {t1=fabs(lat)-fabs(sat_lat);
                t1=fabs(t1);
                t2=(180-sat_lon)+(180-fabs(lon));
                td=t1+t2;
                if(td<total_dif)
                        {total_dif=td;
                         ansr=i;
                        }
                }
        }
 }/*closes lat<0 */
}/*closes fabs<18*/
 }/*closes loop*/
 return(ansr);
}/*end get_sat*/

 /* The Adj_matrix function is used to determine the adjancey matrix for the
satellite system. It constructs an n x n matrix of zeroes and ones.  Elements
of this matrix are either a zero or a one.  A one in position ij indicates
that there is an edge, or cross-link, between satellites i and j When
called the function requires two intergers- the number of orbital planes and
the number of satellites per plane. The function will ask the satellite number
within a plane that is not able to maintain cross links to the left and right.
For Iridium there will be 3 satellites in each plane that are not able to
maintain cross-links.  This adjancy matrix will hold as long as the geometry of
the situation is stable, that as long as the list of satellites maintianing
cross link and the list not maintaing cross links is constant. For Iridium, the
geometry remain constant for the first 3 minutes and 30 seconds of a satellite
pass over a ground station.  The second geometry remains constant for the
remainder of the pass of a single satellite over a ground station */
adj_matrix Adj_matrix(int p, int sats)
 { adj_matrix X;
        int i,j,n,x,y,z;
        n=(p*sats)-1;
        cout<<"Enter the first satellite number that has lost L/R link"<<endl;
        cin>>x;
```

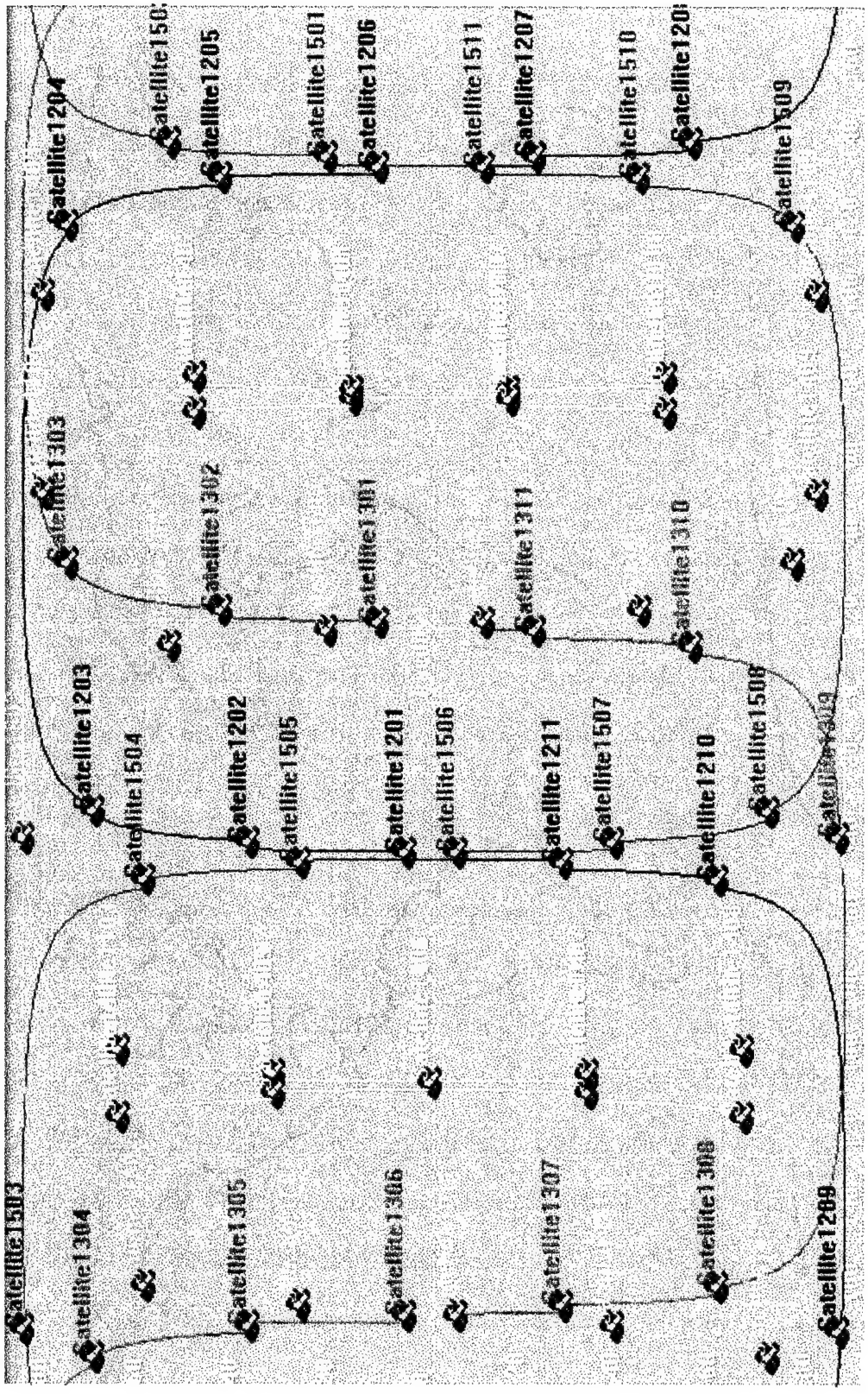
```
        cout<<"Enter the second satellite number that has lost L/R link"<<endl;
        cin>>y;
        cout<<"Enter the final satellite number that has lost L/R link"<<endl;
        cin>>z;
        /* These are used to make sure that satellits with no L/R link are not
        credited with an edge in the adjancey matrix.  1 is subtracted from each
        because C++ arrays start at 0,0 ie if the 1 satellites have no cross link
        than that corresponds to the 0 satellites in the matrix*/
        x=x-1; y=y-1; z=z-1;


        /*set all values of the adjancey matrix equal to zero*/
        for(i=0;i<=n;i++)
          {for(j=0;j<=n;j++)
                X.A[i][j]=0;
          }

/*set the values of A to one for the in-plane ie front and back cross-links*/
        for(i=0;i<=n;i++)
          {for(j=0;j<=n;j++)       /* % is the mod function*/
                {if(i==j && i%11>0 && (i+1)%11>0)
                        {X.A[i][j-1]=1;
                         X.A[i][j+1]=1;
                        }
                 if(i==j && i%11==0)
                  {X.A[i][j+1]=1;
                        X.A[i][j+10]=1;
                  }
                 if(i==j && (i+1)%11==0)
                  {X.A[i][j-1]=1;
                        X.A[i][j-10]=1;
                  }
                 /*set the values for the left and right cross links*/
                 if(i==j && i>10 && i<55 && i!=x+11 && i!=x+22 && i!=x+33 &&
i!=x+44
                        &&i!=y+11 && i!=y+22 && i!=y+33 && i!=y+44 &&
                        i!=z+11 && i!=z+22 && i!=z+33 && i!=z+44)
                  {X.A[i][i-11]=1;
                        X.A[i][i+11]=1;
                  }
                 if(i==j && i<=10 && i!=x && i!=y && i!=z)
                  {X.A[i][i+11]=1;
                        X.A[i][i+55]=1;
                  }
                 if(i==j && i>=55 && i!=x+55 && i!=y+55 && i!=z+55)
```

```
                        {X.A[i][i-11]=1;
                              X.A[i][i-55]=1;
                        }
                       }
                }
        return(X);
}/*end Adj_matrix*/

 /* The Adj_matrix_mid function is used to determine the adjancey matrix forthe
satellite system when the sytem is in a transitional geometry. It constructs
an n x n matrix of zeroes and ones.  Elements
of this matrix are either a zero or a one.  A one in position ij indicates
that there is an edge, or cross-link, between satellites i and j When
called the function requires two intergers- the number of orbital planes and
the number of satellites per plane. The function will ask the satellite number
within a plane that is not able to maintain cross links to the left and right.
For Iridium there will be 4 satellites in each plane that are not able to
maintain cross-links during transitional geometry. Transitional geometry is
considered to occur when a satellite A is sending information to a satellite
b and the geometry changes before satellite A can reach B, that is epsilon
passes while that message is on an intermediate satellite.
*/
adj_matrix Adj_matrix_mid(int p, int sats)
 { adj_matrix X;
        int i,j,n,x,y,z,z2;
        n=(p*sats)-1;
        cout<<"Enter the satellites w/out L/R link for transitional geometry"<<endl;
        cout<<"Enter the first satellite number that has lost L/R link"<<endl;
        cin>>x;
        cout<<"Enter the second satellite number that has lost L/R link"<<endl;
        cin>>y;
        cout<<"Enter the third satellite number that has lost L/R link"<<endl;
        cin>>z;
        cout<<"Enter the final satellite number that has lost L/R link"<<endl;
        cin>>z2;
        /* These are used to make sure that satellits with no L/R link are not
        credited with an edge in the adjancey matrix.  1 is subtracted from each
        because C++ arrays start at 0,0 ie if the 1 satellites have no cross link
        than that corresponds to the 0 satellites in the matrix*/
        x=x-1; y=y-1; z=z-1;  z2=z2-1;


        /*set all values of the adjancey matrix equal to zero*/
        for(i=0;i<=n;i++)
          {for(j=0;j<=n;j++)
```

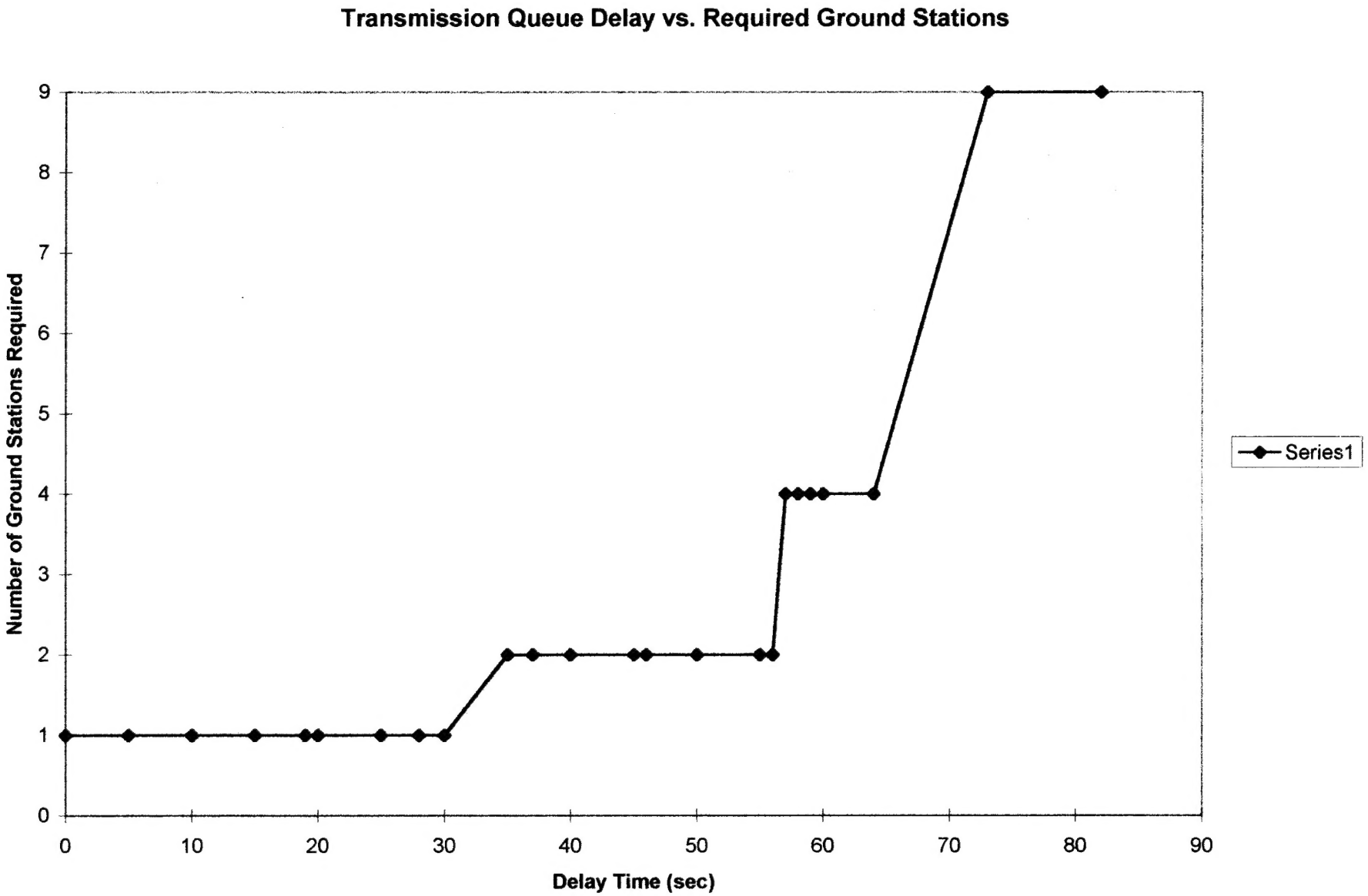
```
                X.A[i][j]=0;
        }

/*set the values of A to one for the in-plane ie front and back cross-links*/
        for(i=0;i<=n;i++)
         {for(j=0;j<=n;j++)        /* % is the mod function*/
                {if(i==j && i%11>0 && (i+1)%11>0)
                                {X.A[i][j-1]=1;
                                 X.A[i][j+1]=1;
                                }
                        if(i==j && i%11==0)
                         {X.A[i][j+1]=1;
                                X.A[i][j+10]=1;
                         }
                        if(i==j && (i+1)%11==0)
                         {X.A[i][j-1]=1;
                                X.A[i][j-10]=1;
                         }
                        /*set the values for the left and right cross links*/
                        if(i==j && i>10 && i<55 && i!=x+11 && i!=x+22 && i!=x+33 &&
i!=x+44
                                &&i!=y+11 && i!=y+22 && i!=y+33 && i!=y+44 &&
                                i!=z+11 && i!=z+22 && i!=z+33 && i!=z+44 &&
                                i!=z2+11 && i!=z2+22 && i!=z2+33 && i!=z2+44)
                         {X.A[i][i-11]=1;
                                X.A[i][i+11]=1;
                         }
                        if(i==j && i<=10 && i!=x && i!=y && i!=z)
                         {X.A[i][i+11]=1;
                                X.A[i][i+55]=1;
                         }
                        if(i==j && i>=55 && i!=x+55 && i!=y+55 && i!=z+55)
                         {X.A[i][i-11]=1;
                                X.A[i][i-55]=1;
                         }
                        }
                }
         return(X);
}/*end Adj_matrix_mid*/

FILE* fopener(void)
        {
                FILE* fp;
                char filename[40], mode[4];
                /* Input the filename and mode */
```

```
        printf("\n Enter a filename you wish to open:");
        gets(filename);
        printf("\n Enter the mode: r to read, w to write:");
        gets(mode);

        /* Try to open the file */
        if ((fp=fopen(filename,mode)) != NULL)
          {printf("\n File %s in mode %s opened", filename, mode);
                return (fp);}
        else
          {printf("\n File %s not able to be opened", filename);
                return(NULL);}
}
```

Appendix 4



Appendix 5


Transmission Queue Delay vs. Required Ground Stations
Number of Ground Stations Required
0
1
2
3
4
5
6
7
8
9
0
10
20
30
40
50
60
70
80
90
Delay Time (sec)
Series1

Ground Station Results for Optimization
For p=0, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=1, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=5, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=10, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=15, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=19, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=20, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=25, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=28, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=30, the number of ground stations was 1
Lat=65.000000 Lon=180.000000

For p=35, the number of ground stations was 2
Lat=65.000000 Lon=180.000000
Lat=65.000000 Lon=160.000000

For p=37, the number of ground stations was 2
Lat=65.000000 Lon=180.000000
Lat=65.000000 Lon=160.000000

For p=40, the number of ground stations was 2
Lat=65.000000 Lon=180.000000
Lat=65.000000 Lon=160.000000

For p=45, the number of ground stations was 2
Lat=65.000000 Lon=180.000000
Lat=35.000000 Lon=-135.000000

For p=46, the number of ground stations was 2
Lat=65.000000 Lon=180.000000
Lat=35.000000 Lon=-135.000000

For p=50, the number of ground stations was 2
Lat=65.000000 Lon=180.000000
Lat=-35.000000 Lon=180.000000

For p=55, the number of ground stations was 2
Lat=65.000000 Lon=180.000000
Lat=-35.000000 Lon=180.000000

For p=56, the number of ground stations was 2
Lat=65.000000 Lon=180.000000
Lat=-35.000000 Lon=180.000000

For p=57, the number of ground stations was 4
Lat=65.000000 Lon=180.000000
Lat=-5.000000 Lon=180.000000
Lat=-20.000000 Lon=180.000000
Lat=65.000000 Lon=75.000000

For p=58, the number of ground stations was 4
Lat=65.000000 Lon=180.000000
Lat=-5.000000 Lon=180.000000
Lat=-20.000000 Lon=180.000000
Lat=65.000000 Lon=75.000000

For p=59, the number of ground stations was 4
Lat=65.000000 Lon=180.000000
Lat=-5.000000 Lon=180.000000
Lat=-20.000000 Lon=180.000000
Lat=65.000000 Lon=75.000000

For p=60, the number of ground stations was 4
Lat=65.000000 Lon=180.000000
Lat=-5.000000 Lon=180.000000
Lat=-20.000000 Lon=180.000000
Lat=65.000000 Lon=75.000000

For p=64, the number of ground stations was 4
Lat=65.000000 Lon=180.000000
Lat=-5.000000 Lon=180.000000
Lat=-20.000000 Lon=180.000000
Lat=65.000000 Lon=75.000000

For p=73, the number of ground stations was 9
Lat=65.000000 Lon=180.000000
Lat=35.000000 Lon=160.000000
Lat=0.000000 Lon=105.000000
Lat=-45.000000 Lon=-50.000000
Lat=-35.000000 Lon=-10.000000
Lat=65.000000 Lon=60.000000
Lat=-20.000000 Lon=-75.000000
Lat=65.000000 Lon=130.000000
Lat=-5.000000 Lon=180.000000

For p=82, the number of ground stations was 9
Lat=65.000000 Lon=180.000000
Lat=35.000000 Lon=160.000000
Lat=0.000000 Lon=105.000000
Lat=-45.000000 Lon=-50.000000
Lat=-35.000000 Lon=-10.000000
Lat=65.000000 Lon=60.000000
Lat=-20.000000 Lon=-75.000000
Lat=65.000000 Lon=130.000000
Lat=-5.000000 Lon=180.000000